Independent spanning trees in Eisenstein–Jacobi networks

Zaid Hussain1 · Hosam AboElFotoh1 · Bader AlBdaiwi1

Accepted: 31 January 2022 / Published online: 28 February 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
Spanning trees are widely used in interconnection networks for routing, broadcasting, fault-tolerance, and securely delivering messages, as well as parallel and distributed computing. The problem of efficiently finding a maximal set of independent spanning trees (ISTs) is still open for arbitrary (general) interconnection topologies (graphs). The first contribution of this paper is presenting two efficient methods that solve the problem in Eisenstein–Jacobi (EJ) networks. The first constructs three Edge-Disjoint Node-Independent Spanning Trees (EDNISTs), whereas the second constructs six ISTs but not edge disjoint. Both constructions have time complexity \(O(n)\), where \(n\) is the number of nodes. The second contribution is providing a distributed fault-tolerant routing algorithm based on the constructed trees with time complexity \(O(1)\) and communication complexity \(O(|P|)\), where \(P\) is the routing path. The experimental results measure the maximum and the average of all maximum number of communication steps between the root node and all other nodes among all ISTs with the consideration of all possible faulty nodes. In addition, the comparison shows that the proposed method outperforms the state-of-the-art method in broadcasting in terms of node coverage in the presence of failures.

Keywords Eisenstein–Jacobi Network · Independent spanning Trees · Fault-tolerant · Routing

Zaid Hussain
zhussain@cs.ku.edu.kw
Hosam AboElFotoh
hosam@cs.ku.edu.kw
Bader AlBdaiwi
bdaiwi@cs.ku.edu.kw

1 Computer Science Department, Kuwait University, Kuwait, Kuwait
1 Introduction

Independent spanning trees (ISTs) play an important role in the reliability of interconnection networks [1, 2]. They are widely used to broadcast messages and to obtain routing paths between nodes in a network. For example, given a regular network of degree $d$, we can tolerate several faulty nodes by constructing $d$ independent spanning trees so that the network will still be connected even with the existence of $d - 1$ faulty nodes. In addition, independent spanning trees are used to securely deliver a message to the destination node [3, 4]. For instance, a message can be sliced into $d$ parts where each part travels in distinct path until all parts reach the destination node where they get assembled. The problem of finding a maximal set of independent spanning trees in networks is known to be NP-Hard [5] and the problem is open for general graphs with connectivity greater than 4 [4].

Over the past years, independent spanning trees have been studied in different types of interconnection networks. For instance, the construction of two completely independent spanning trees in any torus network and in the Cartesian product of any 2-connected graphs is investigated in [6]. In [7], the solution of ISTs in multidimensional torus network is given such that it can determine the parent of each node in a network for a certain IST based on the given node address. Thus, their proposal can be implemented in parallel and distributed systems easily. The construction of edge-disjoint spanning trees in Cartesian product networks has been studied in [8]. It also can be applied to construct the maximum numbers of edge-disjoint spanning trees in hypercubes, Torus, generalized hypercubes, mesh connected trees, hyper Petersen networks, but not in Eisenstein–Jacobi (EJ) networks. Also, it only considered the edge-disjoint and did not discuss the findings for node-independent spanning trees. Further, an optimal ISTs in Cartesian product of hybrid graphs has been investigated in [9], and the provided construction is generalized to other networks. Their results consider $k$-connected graphs with at most $k$ independent spanning trees, where $k \leq 4$. Additionally, the optimal independent spanning trees in the hypercubes are presented in [10]. Further, in [11], the authors discussed a non-recursive and fully parallelized approach to construct ISTs rooted at an arbitrary node on Möbius cubes, and the same problem has been solved for parity cubes in [12]. In addition, in [13], the authors presented a common method for constructing ISTs on bijective connection networks based on $V$-dimensional-permutation technique. Furthermore, building independent spanning trees on twisted cubes has been studied in [14]. And, a simple parallel construction algorithm on twisted cubes has been provided in [15]. In [16], the concept of shortest path routing is used to construct ISTs in generalized recursive circulant graphs. Additionally, a parallel construction of ISTs on highly scalable datacenter networks has been proposed in [17], where the nodes’ parent can be easily found by simple computation based on the node’s address. Also, in [18], the authors constructed the edge-disjoint Hamiltonian cycles in $n$-dimensional augmented cubes ($AQ_n$) to find the maximum number of completely ISTs in data center network based on $AQ_n$. A non-recursive parallel algorithm for building
ISTs in bubble-sort networks is given in [19], where each node can determine its parent in constant time. There are some research studies on building ISTs in other networks such as (but not limited to): crossed cubes [20], locally twisted cubes [21], folded hypercubes [4, 22], enhanced hypercubes [23], pancake networks [24], and transpose networks [25].

Our previous studies on independent spanning trees include the following. In [26], the two edge-disjoint node-independent spanning trees have been constructed for dense Gaussian networks. Further, in [27, 28], the construction and parallel construction of four independent spanning trees were presented such that the edges are not disjoint. Both studies have tree height \(2k\), where \(k\) is the diameter of the network. Lately, a parallel construction algorithm and its evaluation for edge-disjoint node-independent spanning trees in dense Gaussian networks was introduced in [27].

As for EJ networks, the problem of finding ISTs is still unsolved. EJ networks were proposed in [29, 30]. They are generated based on EJ integers [31]. EJ networks are symmetric 6−regular networks and they are generalizations of the hexagonal mesh topology presented in [32, 33]. One of the advantages of these type of networks is that they are used as a new method for constructing some classes of perfect codes that are used to solve the problem of finding the perfect dominating set [31, 30]. In addition, there are some studies on the applications of EJ networks such as routing, broadcasting, and Hamiltonian cycles [29, 34]. In the next sections, we provide the construction method for edge-disjoint node-independent spanning trees and node-independent spanning trees in EJ networks. In the proposed method, each node’s parent and child nodes in the network can be determined based on the given node address. Thus, our proposed method can be easily implemented in parallel and distributed systems. Furthermore, since EJ networks are symmetric networks, then the proposed construction method can start on any node in the network by mapping the selected node to node 0, and mapping all other nodes accordingly.

The three main contributions of this paper are as follows. First, we present a construction of three edge-disjoint node-independent spanning trees (EDNIST) in EJ networks. Second, we introduce a construction of six node-independent spanning trees (IST) in EJ networks. Note that both constructions can also be applied in hexagonal networks. Third, we develop routing algorithms based on the constructed trees used in fault-tolerant point-to-point routing, fault-tolerant broadcasting, or secure message distributions. The designed algorithms are unified in the sense that they can be initiated from any node in an EJ network due to the network topology symmetry and node transitivity.

Throughout this paper, the terms vertices and nodes are used interchangeably; similarly for edges and links, and graph and network. The rest of this paper is organized as follows. We review some graph theory terminologies and briefly describe the EJ networks in Sect. 2. Section 3 introduces the construction of node-independent spanning trees and edge-disjoint node-independent spanning trees in EJ networks. Section 4 presents the routing algorithm. The simulation results are described in Sect. 5. Finally, the paper concludes in Sect. 6.
2 Background

Based on graph theory, some definitions and properties of graph are reviewed in this section. In addition, we briefly describe the topological properties of EJ networks.

Given a graph $G(V, E)$ such that $V$ is the set of $|V|$ vertices and $E$ is the set of $|E|$ edges. An edge is a direct connection between two vertices denoted as $(u, v)$, such that $u, v \in V$. A sequence of connected edges are called path. That is, a path $P(s, d)$ of length $|P(s, d)| = n$ from vertex $s$ to vertex $d$ in $G$ is a sequence of connected edges $(s, x_1), (x_1, x_2), \ldots, (x_n, d)$ where the intermediate vertices are distinct. Two paths $P_1(u, v)$ and $P_2(u, v)$ are said to be independent if their intermediate set of vertices are disjoint. A tree $ST(V', E')$ that is a subgraph of $G(V, E)$ where $V' \subseteq V$ and $E' \subseteq E$ is called spanning tree when it contains all the vertices of $G$, i.e., $V' = V$. Two or more spanning trees $ST_j$, for $j \geq 1$, rooted at vertex $r$ are called independent spanning trees if $\bigcap_{j=1}^{t}(P_{ST_j}(r, u) \setminus \{r, u\}) = \phi$ for $u \in V$, where $P_{ST_j}(r, u)$ is a path from $r$ to $u$ in the $j^{th}$ spanning tree, and $t$ is the maximum number of ISTs in $G$. Further, the trees which have their edge sets pairwise disjoint are called edge-disjoint node-independent spanning trees. That is, for all trees $ST_j(V, E_j)$ for $j = 1, 2, \ldots, t$, $t$ is the maximum number of EDNISTS in $G$, we have $E'_j \cap E'_j = \phi$ for all $p \neq q$ such that $1 \leq p \leq n$ and $1 \leq q \leq n$. In addition, when each IST has a distinct root node, i.e. the root nodes differ in all trees, then the problem is called completely independent spanning trees (CISTs). In other words, two or more spanning trees $ST_j$, for $j \geq 1$, rooted at vertex $r_j$ are called completely independent spanning trees if $\bigcap_{j=1}^{t}(P_{ST}(r_j, u) \setminus \{r_j, u\}) = \phi$ for $u \in V$, where $P_{ST_j}(r_j, u)$ is a path from $r_j$ to $u$ in the $j^{th}$ spanning tree, $t$ is the maximum number of CISTs in $G$, and $r_j$ is distinct in all trees. Further, when the average path length of the trees is minimum, then the ISTs are called optimal ISTs [10].

In graph $G$, the distance between the two vertices $u$ and $v$ is the number of edges along the shortest path $P(u, v)$ (the path with minimum length over all possible paths between $u$ to $v$). The diameter $k$ of the graph is known as the shortest distance between two farthest vertices in graph $G$. Further, for $j \geq 1$, we define the maximum number of communication steps between the root node $r$ and all other nodes $u \in V$ in the spanning tree $ST_j$, as $ST_{MAX} = \max |P_{ST_j}(r, u)|$. The maximum of maximum number of communication steps among all trees is defined as $\text{max}(ST_{MAX})$ and the average of the maximum number of communication steps among all trees is defined as $\text{average}(ST_{MAX})$, for $j \geq 1$.

Eisenstein–Jacobi networks [29] are based on EJ integers [31, 30]. EJ networks can be modeled on planar graphs as a graph $EJ\alpha(V, E)$. They are generated by $\alpha = a + b \rho$ such that $0 \leq a \leq b$, where $V = \mathbb{Z}[\rho]_\alpha$ is the vertex set modulo $\alpha$, and $E = \{(A, B) \in V \times V : (A - B) \equiv \pm 1, \pm \rho, \pm \rho^2 \mod \alpha\}$ is the edge set. The set of Eisenstein–Jacobi integers $\mathbb{Z}[\rho]$ is defined as $\mathbb{Z}[\rho] = \{x + y\rho \mid x, y \in \mathbb{Z}\}$, where $\rho = (1 + i\sqrt{3})/2$, and $i = \sqrt{-1}$. It is known that $\mathbb{Z}[\rho]$ is a Euclidean domain. The norm of EJ integer $\alpha = a + b \rho$ is given by $N(\alpha) = a^2 + b^2 + ab$ [29], which is the total number of the distinct vertices in the network under the residue class modulo $\alpha$. It can be seen that $\rho^2 = \rho - 1$, $\rho^3 = -1$, $\rho^4 = -\rho$, $\rho^5 = 1 - \rho$, and $\rho^6 = 1$. 
The EJ networks are regular symmetric networks of degree six since each node in
the EJ network has six neighbors. The nodes in the network are addressed by \(x + y\). Two
nodes in the network are adjacent, if and only if there is an edge between them,
i.e., the distance between them is 1.

The distance distribution in the network is based on the distance of the nodes
from the center node, usually node 0. That is, it is the number of nodes at distance
\(t\) from node 0, where \(t > 0\). EJ networks are called dense EJ networks when they
contain a maximum number of nodes at distance \(k\) where \(k\) is the network’s diam-
eter. Usually, their generator is \(a = a + b\rho\) such that \(b = a + 1\). Thus, the number
of nodes at distance \(t\) is 1 or 6\(t\), in respective order, for \(t = 0\) or \(t > 0\). For instance,
given \(a = 3 + 4\rho\), the number of nodes at distances 0, 1, 2, and 3 are 1, 6, 12, and
18 nodes, respectively. In detail, assume node 0 is the center node, to generate node
1 (or \(a\)) we add 1 (or \(\rho\)) to node 0 to get node 1 + 0\(\rho\) (or 0 + \(\rho\)), which is 1 (or \(\rho\))
mod \(a\). The white nodes in Fig. 1 illustrate the node distribution in the EJ network
generated by \(a = 3 + 4\rho\). It can be concluded that the diameter of dense EJ networks
is \(k = a\) and the number of nodes \(d(t)\) at distance \(t\) is:

\[
d(t) = \begin{cases} 
1 & \text{if } t = 0 \\
6t & \text{if } 1 \leq t \leq k 
\end{cases}
\]

\(\text{Fig. 1 EJ Network generated by } a = 3 + 4\rho \text{ with dotted lines as wraparound edges. The gray nodes are the tiled nodes.}\)
There are two types of links in the EJ networks: regular and wraparound links. The regular link connects two neighboring nodes within the same tile of the network, whereas the wraparound links connect two neighboring nodes located in different tiles of the network. The wraparound links can be recognized either by tiling or by modulo operation. By tiling, we mean that placing the EJ network at the origin of a grid and considering it as a basic EJ network with its center node is 0. After that, the tiles are created by copying the basic EJ network and placing its copies around it. By modulo operation, we use the \( \text{mod} \) operator after adding \( \pm 1 \), \( \pm \rho \), or \( \pm \rho^2 \) to the EJ integers to get the corresponding nodes in the basic EJ network. Figure 1 illustrates the EJ network generated by \( \alpha = 3 + 4\rho \) with dotted lines as wraparound links. Furthermore, the shaded (gray) nodes are part of the tiled EJ networks connected to the basic EJ network, and the rest of the tiled nodes are omitted. The nodes in different tiles of the network are represented in different gray colors.

**Example 1** Consider the node \( 3\rho \) in Fig. 1. The node \( 3\rho \) is connected to node \( 1 + 3\rho \), which its corresponding node is \( -2 - \rho \) in the basic EJ network, through \(+1\) edge. That is, the resultant of adding \(+1\) to node \( 3\rho \) and then taking the \( \text{mod} \) \( \alpha \) is node \( -2 - \rho \). Similarly, the \( +\rho \) and \( +\rho^2 \) edges connect the node \( 3\rho \) to nodes, in respective order, \( 4\rho \) and \( 3\rho + \rho^2 \), which their corresponding nodes in the basic EJ network are \( -3 \) and \( -3\rho^2 \), respectively.

### 3 Independent spanning trees construction

#### 3.1 Edge-disjoint node-independent spanning trees

Partitioning the network is helpful in finding edge-disjoint node-independent spanning trees. Given the EJ network generated by \( \alpha = a + b\rho \) such that \( b = a + 1 \) and represented as \( G(V, E) \), the set \( V \) can be partitioned into subsets as illustrated in Fig. 2. Let \( c = 0, 2, 4 \) for tree \( t = 1, 2, 3 \), respectively, and for \( d = 1, 2, 3, 4, 5, 6 \) such that \( |x| + |y| = k \) where \( k \) is the diameter of the network. Then, the subsets are as follows (all the powers of \( \rho \) are modulo 6):

---

**Fig. 2** EDNIST partitions

---
\[ B_d = \{x \rho^{j-1} + y \rho^j \mid x > 0, y = 0, j = d + c \}. \quad (2) \]

\[ T_d = \{x \rho^{j-1} + y \rho^j \mid x > 0, y > 0, j = d + c \}. \quad (3) \]

\[ S_2 = \{x \rho^{j-1} + y \rho^j \mid x = a, y = 0, j = 2 + c \}. \quad (4) \]

\[ S_4 = \{x \rho^{j-1} + y \rho^j \mid x = a, y = 0, j = 4 + c \}. \quad (5) \]

\[ L_4 = \{x \rho^{j-1} + y \rho^j \mid x > 0, y = 1, j = 4 + c \}. \quad (6) \]

\[ L_6 = \{x \rho^{j-1} + y \rho^j \mid x > 0, y = 1, j = 6 + c \}. \quad (7) \]

\[ B_2 \setminus S_2 = \{x \rho^{j-1} + y \rho^j \mid 0 < x < k, y = 0, j = 2 + c \}. \quad (8) \]

\[ B_4 \setminus S_4 = \{x \rho^{j-1} + y \rho^j \mid 0 < x < k, y = 0, j = 4 + c \}. \quad (9) \]

\[ T_4 \setminus L_4 = \{x \rho^{j-1} + y \rho^j \mid x > 0, y > 1, j = 4 + c \}. \quad (10) \]

\[ T_6 \setminus L_6 = \{x \rho^{j-1} + y \rho^j \mid x > 0, y > 1, j = 6 + c \}. \quad (11) \]

**Lemma 1** The subsets described in formulas (2)–(11) are disjoint.

**Proof** Let \( V \) be the set of subsets defined in formulas (2)–(11) and illustrated in Fig. 2. That is, \( V = \{B_1, T_1, (B_2 \setminus S_2), S_2, T_2, B_3, T_3, (B_4 \setminus S_4), S_4, L_4, (T_4 \setminus L_4), B_5, T_5, B_6, L_6, (T_6 \setminus L_6) \}. \) Based on the definition of the subsets, for any two subsets \( X, Y \in V, X \neq Y, X \cap Y = \emptyset. \)

**Lemma 2** The subsets described in formulas (2)–(11) contain all nodes in the network.

**Proof** Based on formulas (2)–(11) and given the norm as a total number of nodes in the network, \( N(\alpha) = a^2 + b^2 + ab \), then for \( \alpha = k + (k + 1)\rho \) we get \( N(\alpha) = 3k^2 + 2k + 1 \). It is obvious that \( |B_d| = k \) for \( d = 1, 3, 5, 6 \). Thus, we got a total of \( 4k \). In addition, \( |S_2| = |S_4| = 1, |B_2 \setminus S_2| = |B_4 \setminus S_4| = k - 1, |L_4| = |L_6| = k - 1 \). Further, \( |T_4 \setminus L_4| = \sum_{i=1}^{k-1} \sum_{j=1}^{k-i} 1 = \sum_{i=1}^{k-1} (k - i) = 1/2(k - 1)k \) for \( d = 1, 2, 3, 5, 6 \). That is, a total of \( 2(k - 1)k \). Finally, we have \( |T_6 \setminus L_6| = |T_6 \setminus L_6| = 1/2(k - 1)k - (k - 1) \). Thus, \( B_d \cup T_d \cup S_2 \cup S_4 \cup L_4 \cup L_6 \cup (B_2 \setminus S_2) \cup (B_4 \setminus S_4) \cup (T_4 \setminus L_4) \cup (T_6 \setminus L_6) \cup \{0\} \) (including node 0) is equal to the set \( V \), which is the set of nodes in the network. We conclude that,
\[ 4|B_d| + 4|T_d| + |S_2| + |S_4| + |L_4| + |L_6| + |B_2 \setminus S_2| + |B_4 \setminus S_4| + |T_4 \setminus L_4| + |T_6 \setminus L_6| + +|\{0\}| = 3k^2 + 3k + 1 = N(\alpha) \) (excluding \( B_2, B_4, T_4, \) and \( T_6 \)).

Table 1 illustrates the parent and child nodes in the spanning tree for a given node belonging to a certain set. It can be used to construct the spanning trees for a given EJ network.

**Example 2** Given the EJ network generated by \( \alpha = 4 + 5\rho \) and node \( v = 1 + \rho \). In the first spanning tree, since \( v \in T_1 \), its parent is node 1 and its child is node 1 + 2\( \rho \).

**Lemma 3** Let \( ST_{ED} \) be a set of edge disjoint node independent spanning trees in the EJ network generated by \( \alpha = a + b\rho \), where \( b = a + 1 \), then \( |ST_{ED}| \leq 3 \).

**Proof** The total number of nodes in the EJ network generated by \( \alpha = a + b\rho \) is known as \( N(\alpha) = a^2 + b^2 + ab \). In case of \( b = a + 1 \), the total number of nodes is \( 3a^2 + 3a + 1 \) and the total number of undirected edges is \( 9a^2 + 9a + 3 \). Since the spanning trees are edge disjoint, then each spanning tree \( ST_{ED} \) must have exactly \( 3a^2 + 3a \) undirected edges. Thus, it follows that \( |ST_{ED}| \leq 3 \).

**Lemma 4** The first spanning tree is connected.

**Proof** Based on formulas (2)–(11), consider \( j \) values with \( c = 0 \). Let \( ST_{ED}(V_1, E_1) \) represent the first edge disjoint node independent spanning tree where \( V_1 \subseteq V \) and \( E_1 \subseteq E \) are the set of nodes and edges in \( ST_{ED} \), respectively. Based on Lemma 3, we have \( |E_1| = 3a^2 + 3a = |V_1| - 1 \). Further, Table 2 shows the path from the source node \( S = 0 \) to all other nodes in the network using tree \( ST_{ED} \). As noted in Table 2, the paths are described by a word on the alphabet \( \{-1, 1, -\rho, \rho, -\rho^2, \rho^2\} \) where the symbols denote the direction of the edges to be passed. The number of steps is represented as \((direction)^{steps}\). We conclude that \( ST_{ED} \) is connected.
Example 3 In the first spanning tree, let \( S = 0 \) and \( D = \rho^4 + 3\rho^5 \) (which is \( D = -1 - 4\rho^2 \)) where \( x = -1 \) and \( y = -4 \), then \( D \in B_6 \cup T_5 \cup B_5 \). Thus, using Table 2, the steps are \((−\rho^2)^4(−1)^2\). That is, \( D \) can be reached by going 1 step along direction 1, then 4 steps along direction \( −\rho^2 \), and finally 2 steps along direction \( −1 \).

Lemma 5 The second and third spanning trees can be obtained by rotating the first spanning tree.

Proof Based on Lemmas 1 and 2, and Table 1, since the network is symmetric, then it is sufficient to prove that the obtained second and third spanning trees are connected by following Lemma 4, but with different \( j \) values when \( c = 2, 4 \) for formulas (2)–(11).

Theorem 1 \( ST_{ED} \), for \( t = 1, 2, 3 \), are edge disjoint node independent spanning trees.

Proof Based on Lemmas 1–5, and Tables 1 and 2, let \( ST_{ED_t}(E) \) be the set of undirected edges for spanning tree \( t \). Thus, we get \( ST_{ED_1}(E) \cap ST_{ED_2}(E) = \phi, t, t' \in \{1, 2, 3\}, t \neq t' \). We conclude that all trees are edge disjoint node independent spanning trees.

Lemma 6 The depth of all trees \( ST_{ED} \), for \( t = 1, 2, 3 \), is \( 2k + 2 \).

Proof The proof is provided for tree \( ST_{ED_1} \); The same proof can be applied to the other trees accordingly. Based on Lemma 2 and Table 2, the longest path in tree \( ST_{ED_1} \) starting from node 0 is \( 2k + 2 \), which leads to node \( −k\rho \) or to node \( k\rho^2 \). Further, the last set in Table 2 is \( B_6 \cup T_5 \cup B_5 \) which has a maximum steps of \( \max|x| + \max|y| + 1 + 1 = k + k + 2 = 2k + 2 \).
Independent spanning trees in Eisenstein–Jacobi networks

Example 4 Consider the EJ network generated by \( \alpha = 4 + 5\rho \). The first EDNIST can be constructed using Table 1 by determining the parent and child nodes of any selected node in the network, except the root node. After constructing the first ENDIST as shown in Fig. 3a, then we can use Lemma 5 to rotate the first EDNIST to obtain the second and third EDNIST as illustrated in Figs. 3b and 3c, respectively.

3.2 Node-independent spanning trees

Given the EJ network generated by \( \alpha = a + b\rho \) such that \( b = a + 1 \) and represented as \( G(V, E) \), the set \( V \) can be partitioned into subsets as shown in Fig. 4. The subsets are described as follows. Let \( c = t - 1 \) for tree \( t = 1, 2, 3, 4, 5, 6 \), \( d = 1, 2, 3, 4, 5, 6 \), and all the powers of \( \rho \) are modulo 6. In addition, Let \( |x| + |y| = k \) where \( k \) is the network diameter, then:

\[
B_d = \{ x\rho^{j-1} + y\rho^j \mid x > 0, y = 0, j = d + c \}. \tag{12}
\]

\[
T_d = \{ x\rho^{j-1} + y\rho^j \mid x > 0, y > 0, j = d + c \}. \tag{13}
\]

\[
S = \{ x\rho^{j-1} + y\rho^j \mid x = 1, y = 0, j = 5 + c \}. \tag{14}
\]
\[ B_5 \setminus S = \{ x\rho^{j-1} + y\rho^j \mid x > 1, y = 0, j = 5 + c \}. \]  
\[ L_3 = \{ x\rho^{j-1} + y\rho^j \mid x > 0, y = 1, j = 3 + c \}. \]  
\[ L_4 = \{ x\rho^{j-1} + y\rho^j \mid x > 0, y = 1, j = 4 + c \}. \]  
\[ T_3 \setminus L_3 = \{ x\rho^{j-1} + y\rho^j \mid x > 0, y > 1, j = 3 + c \}. \]  
\[ T_4 \setminus L_4 = \{ x\rho^{j-1} + y\rho^j \mid x > 0, y > 1, j = 4 + c \}. \]  

**Lemma 7** The subsets described in formulas (12)–(19) are disjoint.

**Proof** Let \( V \) be the set of subsets defined in formulas (12)–(19) and illustrated in Fig. 4. That is, \( V = \{ B_1, T_1, B_2, T_2, B_3, L_3, (T_3 \setminus L_3), B_4, L_4, (T_4 \setminus L_4), (B_5 \setminus S), S, T_5, B_6, T_6 \} \). Based on the definition of the subsets, for any two subsets \( X, Y \in V, X \neq Y, X \cap Y = \emptyset \).

**Lemma 8** The subsets described in formulas (12)–(19) contain all nodes in the network.

**Proof** Based on formulas (12)–(19) and given the norm as a total number of nodes in the network, \( N(\alpha) = a^2 + b^2 + ab \), then for \( \alpha = k + (k + 1)\rho \) we get \( N(\alpha) = 3k^2 + 2k + 1 \). It is obvious that \( |B_d| = k \) for \( d = 1, 2, 3, 4, 6 \). Thus, we got a total of 5k. In addition, \( |S| = 1, |B_5 \setminus S| = k - 1, |L_3| = k - 1, |L_4| = k - 1 \). Further, \( |T_d| = \sum_{i=1}^{k-1} \sum_{j=1}^{k-i} 1 = \sum_{i=1}^{k-1} (k - i) = 1/2(k - 1)k \) for \( d = 1, 2, 5, 6 \). That is, a total of \( 2(k - 1)k \). Finally, we have \( |T_3 \setminus L_3| = |T_4 \setminus L_4| = 1/2(k - 1)k - (k - 1) \). Thus, \( B_d \cup T_d \cup S \cup (B_5 \setminus S) \cup L_3 \cup L_4 \cup (T_3 \setminus L_3) \cup (T_4 \setminus L_4) \cup \{ 0 \} \) (including node 0) is equal to the set \( V \), which is the set of nodes in the network. We conclude that, \( 5|B_d| + 4|T_d| + |S| + |(B_5 \setminus S)| + |L_3| + |L_4| + |(T_3 \setminus L_3)| + |(T_4 \setminus L_4)| + |\{ 0 \}| = 3k^2 + 3k + 1 = N(\alpha) \) (excluding \( |B_5|, |T_3|, \) and \( |T_4| \)).

Similar to Sect. 3.1, the node independent spanning trees can be constructed using on Table 3, which provides the parent and child nodes for a given node in a certain set.

**Example 5** Given the EJ network generated by \( \alpha = 4 + 5\rho \) and node \( v = 4\rho \). In the first spanning tree, since \( v \in B_2 \) then its parent is node \( 4\rho - 1 \) and it has no child.

**Lemma 9** Let \( ST \) be a set of node independent spanning trees in the EJ network generated by \( \alpha = a + b\rho \), where \( b = a + 1 \), then \( |ST| \leq 6 \).

\( \copyright \) Springer
Example 6 In the first spanning tree, let \( S = 0 \) and \( D = \rho^4 + 3\rho^5 \) (which is \( D = 3 - 4\rho \)) where \( x = 3 \) and \( y = -4 \), then \( E \in \{B_1 \cup S\} \cup S \cup T_5 \cup B_6 \). Thus, the steps are \((1)^4(\rho)^1(1)^3\). That is, \( D \) can be reached by going 4 steps along direction 1, then 1 step along direction \( \rho \), and finally 3 steps along direction 1.
Lemma 11 The second, third, fourth, fifth, and sixth node independent spanning trees can be obtained by rotating the first node independent spanning tree.

Proof Based on Lemmas 7 and 8, and Table 3, since the network is symmetric, then it is sufficient to prove that the obtained second, third, fourth, fifth, and sixth node independent spanning trees are connected by following Lemma 10, but with different $j$ values when $c = t - 1$ for $t = 2, 3, 4, 5, 6$ as described in formulas (12)–(19).

Theorem 2 $ST_t$, for $t = 1, 2, 3, 4, 5, 6$, are node independent spanning trees.

Proof Based on Lemmas 7–11, and Tables 3 and 4, let $ST_t(E)$ be the set of directed edges for the spanning tree $t$. Thus, we get $ST_t(E) \cap ST_{t'}(E) = \emptyset, t, t' \in \{1, 2, 3, 4, 5, 6\}, t \neq t'$. That is, each directed edge is used once among all trees. We conclude that all trees are node independent spanning trees.

Lemma 12 The depth of all trees $ST_t$, for $t = 1, 2, 3, 4, 5, 6$, is $2k + 1$.

Proof The proof is provided for tree $ST_1$. The same proof can be applied to the other trees accordingly. Based on Lemma 8 and Table 4, the longest path in tree $ST_1$ starting from node 0 is $2k + 1$, which leads to nodes $c\rho$ or to nodes $-c\rho^2$, where $1 \leq c \leq k$.

Example 7 The same argument explained in Example 4 can be applied to construct the 3 ISTs in the EJ network generated by $\alpha = 4 + 5\rho$. Instead, Table 4, Table 3, and Lemma 11 are used for the construction. All constructed ISTs are illustrated in Fig. 5.

3.3 Construction algorithm

In this subsection, we present a unified algorithm to construct both EDNISTs and ISTs in a given graph of the EJ network.

Based on Sect. 3.1 (or 3.2), the EDNISTs (or ISTs) can be constructed as illustrated in Algorithm 1 and as follows. For each tree construction, we perform: (1) The value of $c$ is set accordingly to be used in formulas (2)–(11) (or (12)–(19)) and Table 1 (or Table 3). (2) For each node $v$ in the network we iterate from $d = 1$ to $d = 6$ based on the maximum number of subsets (partitions) index. (3) Based on formulas (2)–(11) (or (12)–(19)), we determine which subset that $v$ belongs to. (4) We match the determined subset with Table 1 (or Table 3) to obtain the parent and child nodes to be linked to $v$. Algorithm 1 takes a graph and the type of the tree to be constructed as inputs.
Since $|V| = n$ is the total number of nodes, then each EDNIST (or IST) takes $n$ steps to be constructed. It is obvious that the time complexity of Algorithm 1 is $O(n)$ since it contains 3 nested loops. The first, second, and third loops take, in respective order, 3 (or 6 for IST) in line 2, $n$ as a total number of nodes in line 8, and 6 in line 9. That is, the number of iterations is $3 \times n \times 6 = 18n$ (or $6 \times n \times 6 = 36n$). Thus, the time complexity to construct 3 EDNISTs (or 6 ISTs) is $O(n)$.

**Algorithm 1** Construct($G(V,E)$, type)

1. Let $c = [0, 2, 4]$ be an array with starting index 1
2. for $t = 1$ to 3 (or 6 for IST) do
3. if $type = EDNIST$ then
4. Use $c[t]$ as $c$ in formulas (2)-(11)
5. else
6. $c = t - 1$
7. end if
8. for $v \in V$ do
9. for $d = 1$ to 6 do
10. Examine $v$ with formulas (2)-(11) (or (12)-(19) for IST) to determine $v$’s subset
11. Match the determined subset with Table 1 (or Table 3 for IST) to get the parent and child nodes of $v$
12. Connect $v$ to parent and child nodes
13. end for
14. end for
15. end for

Fig. 5 ISTs in EJ with $\alpha = 4 + 5\rho$. a First, b Second, c Third, d Fourth, e Fifth, and f Sixth ISTs
4 Routing

In this section, we present the algorithm used to route the messages in the trees constructed in Sects. 3.1 and 3.2. The algorithm uses Table 2 and Table 4 to determine the link in the current node to be used for sending/forwarding the messages in the constructed EDNIST and IST, respectively.

Algorithm 2 describes the procedures to be taken at the source node as follows. In line 1, since Tables 2 and 4 assume the source node is 0 and due to the symmetry of the network, the given source node $S$ is mapped to node 0, and the destination node $D$ is also mapped accordingly. Line 2, obtains the path sequence as tuples consisting of $\text{direction, steps}$ based on the $\text{Subset}$ that the destination node $D$ belongs to. The $\text{direction}$ represents the link to be used in the current node to send/forward the message and the $\text{steps}$ is the number of hops along the given $\text{direction}$. In line 3, the first tuple is obtained to be used to send the message in line 4.

In Algorithm 3, lines 1–4 check whether the message has arrived at the destination node to be consumed. Lines 5–7, checks whether the number of the steps is equal to 0. If so, then it means that there are no more steps in the current given direction. Thus, a tuple is obtained from the current path $P$ sequence, where the remaining tuples will be obtained later on. In line 8, the algorithm sends the message using the link described in $\text{direction}$ and reduces the number of $\text{steps}$ by 1.

The time complexity of Algorithm 2 is computed as follows. Line 1 is a mathematical operation to map $S$ to 0 and $D$ accordingly, which takes $O(1)$. Line 2, looking up a table to match the $D$ with its corresponding $\text{Subset}$ takes $O(1)$ since it only checks 8 (or 7) conditions for each subset in Table 2 (or Table 4). Lines 3–4, each takes $O(1)$. Thus, the total time complexity of Algorithm 2 is $O(1)$. The communication complexity is $O(1)$ since it only sends one message as stated in line 4. Furthermore, the time complexity of Algorithm 3 is $O(1)$ since each line takes constant time. The communication complexity is $O(1)$ per node as stated in line 8.
We conclude that the total routing time complexity is $O(1)$ and the total communication complexity is $O(|P|)$, where $|P|$ is the length of the routing path. The following example illustrates the usage of the routing algorithm.

**Example 8** Let the source node be $S = 0$ and the destination node be $D = -2 - \rho^2$ in the EJ network generated by $\alpha = 4 + 5\rho$. We get $k = 4, x = -2$, and $y = -1$. Based on Algorithm 2, no need to map $S$ because it is 0 and we obtain path $P = \{(1, 3), (-\rho^2, 3)\}$ since $D \in \text{Set} = T_3\{B_4 \setminus S_4\} \cup S_4$. The dir is set to 1 and the steps is set to 3 by calling $P.pop()$, which results in $P = \{(-\rho^2, 3)\}$. After that, based on Algorithm 2, the source node $S$ sends the message $\text{Route}(1, 2, P, 1, -2 - \rho^2, \alpha)$ through link 1 to node 1. Node 1 applies the line 8 in Algorithm 3 and continue sending the message $\text{Route}(1, 1, P, 2, -2 - \rho^2, \alpha)$ to node 2 via link 1. Node 2 applies the line 8 in Algorithm 3 and continue sending the message $\text{Route}(1, 0, P, 3, -2 - \rho^2, \alpha)$ to node 3 via link 1. At node 3, since the steps = 0, then it gets the next tuple by calling $P.pop()$ and sets dir to $-\rho^2$ and steps to 3, after that it continues sending the message $\text{Route}(-\rho^2, 2, P, 3 - \rho^2, -2 - \rho^2, \alpha)$ to node 3 $- \rho^2$ via link $- \rho^2$. The receiving node $3 - \rho^2$ applies line 8 in Algorithm 3 and continues sending the message $\text{Route}(-\rho^2, 1, P, 3 - 2\rho^2, -2 - \rho^2, \alpha)$ to node $3 - 2\rho^2$ via link $- \rho^2$. The receiving node $3 - 2\rho^2$ applies line 8 in Algorithm 3 and continues sending the message $\text{Route}(-\rho^2, 0, P, 3 - 3\rho^2, -2 - \rho^2, \alpha)$ to node $3 - 3\rho^2$ via link $- \rho^2$. Finally, the receiving node $3 - 3\rho^2$ observes that $S = 3 - 3\rho^2 \equiv -2 - \rho^2 = D \ mod \ \alpha$ and receives the message.

Further, the proposed routing algorithm can be used to secure the message distribution. For instance, assume that node $u$ wants to send a message $M$ to node $v$. First, $u$ splits $M$ into $t$ parts, say $m_1, m_2, ..., m_t$, where $t$ is the maximum number of ISTs or EDNISTs. Then, $u$ sends each $m_i, 1 \leq i \leq t$, to $v$ using different spanning trees. At the end, $v$ collects and assembles all the parts to obtain the message $M$. Thus, the message parts $m_i$’s are propagated through the network using different spanning trees where the intermediate nodes have no clue about the content of $M$.

## 5 Experimental results

In this section, we discuss the simulation results. We have used a Python network simulator called NetworkX [35] in the implementation. It is a package used to represent and analyze the networks and the algorithms used in the networks. In simulation, we assumed that each node can send and receive messages simultaneously to all its neighbors.

The algorithm in Sect. 3.2 always constructs 6 trees where the maximum number of steps required to construct the trees is $2k + 2$. Furthermore, we have measured the maximum and the average of all maximum communication steps between the root node and all other nodes in the network among all trees with the following case. (1) no faulty node, (2) all possible locations of one faulty node, (3) all possible locations of two faulty nodes, (4) all possible locations of three
faulty nodes, (5) all possible locations of four faulty nodes, and (6) all possible locations of five faulty nodes. That is, we have developed a brute-force algorithm to simulate all possible locations of faulty nodes. Note that, we did not measure beyond 5 faulty nodes, since in the worst-case scenario, the root node will be pruned from the trees if all its neighbors are faulty. That is, the root node will be isolated from the network and there will be no path to other nodes that can be used to measure the efficiency of the communications.

The network sizes selected in the simulation are $\alpha = 1 + 2 \rho$, $\alpha = 2 + 3 \rho$, $\alpha = 3 + 4 \rho$, $\alpha = 4 + 5 \rho$, $\alpha = 5 + 6 \rho$, $\alpha = 6 + 7 \rho$, $\alpha = 7 + 8 \rho$, $\alpha = 8 + 9 \rho$, and $\alpha = 9 + 10 \rho$. The results of the simulation are illustrated in Tables 5 and 6. Some values are omitted due to hardware resource limitations. Table 5 shows the average maximum number of communication steps in all ISTs using all ports, whereas Table 6 shows the maximum of all maximum number of steps to reach all nodes among all ISTs.

### Table 5 Average maximum number of steps to reach all nodes among all ISTs

| $\alpha$ | $1 + 2 \rho$ | $2 + 3 \rho$ | $3 + 4 \rho$ | $4 + 5 \rho$ | $5 + 6 \rho$ | $6 + 7 \rho$ | $7 + 8 \rho$ | $8 + 9 \rho$ | $9 + 10 \rho$ |
|---------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Lower Bound | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| No Faulty | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 Faulty | 2 | 3.333 | 4.5 | 5.6 | 6.666 | 7.714 | 8.75 | 9.777 | 10.8 |
| 2 Faulty | 2 | 3.529 | 4.852 | 6.061 | 7.208 | 8.32 | 9.409 | 10.481 | 11.542 |
| 3 Faulty | 2 | 3.649 | 5.105 | 6.417 | 7.648 | 8.831 | 9.98 | 11.107 | 12.216 |
| 4 Faulty | 2 | 3.765 | 5.314 | 6.71 | 8.017 | 9.266 | 10.477 |
| 5 Faulty | 2 | 3.899 | 5.512 | 6.971 | 8.339 |
| Upper Bound | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |

### Table 6 Maximum of all maximum number of steps to reach all nodes among all ISTs

| $\alpha$ | $1 + 2 \rho$ | $2 + 3 \rho$ | $3 + 4 \rho$ | $4 + 5 \rho$ | $5 + 6 \rho$ | $6 + 7 \rho$ | $7 + 8 \rho$ | $8 + 9 \rho$ | $9 + 10 \rho$ |
|---------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Lower Bound | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| No Faulty | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 Faulty | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 |
| 2 Faulty | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 |
| 3 Faulty | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 |
| 4 Faulty | 2 | 6 | 8 | 10 | 12 | 14 | 16 |
| 5 Faulty | 2 | 6 | 8 | 10 | 12 |
| Upper Bound | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |

It is observable that the simulation results are consistent with the discussions in Sect. 3.2 where the results are bounded by the lower and upper bounds. The lower bound is $k + 1$, whereas the upper bound is $2k + 2$, which is equal to the tree depth $- 1$. That is, one more step is counted when the last node is trying to communicate to its neighboring nodes.
The simulation measures the required number of communication steps to reach each destination node $D$ from the source node $S = 0$ with no faulty node. In one faulty node, we run the simulation $n$ times, where $n$ is the total number of nodes in the network. In each run, we take one of the nodes down then we measure the required number of communication steps to reach the destination node. That is, in case of one faulty node, for each network size, we have measured the maximum number of communication steps required to reach each node in the network from $S$ with all possible locations of one faulty node. Then, we obtain the average and the maximum of the steps. The same simulation is applied for the cases when all possibilities of 2, 3, 4, and 5 faulty nodes are present in each network size.

Furthermore, since this paper is the first to introduce routing using EDNIST and IST in EJ networks, we are unable to compare our proposed method with other methods related to EDNIST and IST in EJ networks. In comparison with the routing algorithm, the routing in [36, 29] has only one path and does not consider the presence of any faulty node, which leads to non-fault-tolerant routing, whereas the proposed method takes the faulty nodes into consideration with the presence up to 5 faulty nodes. The proposed routing uses EDNIST (or IST) to reach the destination node using 3 (or 6) independent paths. Thus, the proposed method tolerates up to 5 faulty nodes.

On the other hand, we have compared our proposed method with the one-to-all broadcasting developed in [36] (we call it the referenced method). We define the following. (1) The disconnected nodes are the nodes in the network that are unreachable and connected along to the faulty node, i.e., the faulty node is their ancestor. (2) The faulty nodes and the disconnected nodes are considered unreachable nodes. (3) The reachable nodes = total nodes − unreachable nodes. We have computed the broadcasting node coverage, which shows how reliable the network is based on the number of reachable nodes with respect to the total nodes in the network. Table 7 describes the comparison between the proposed method with the presence of 1, 2, 3, 4, and 5 faulty nodes in IST with the referenced method.

From Table 7, it is obvious that the proposed method outperforms the algorithm in [36] in all network sizes with the presence of one faulty node. With the presence of 2, 3, 4, and 5 faulty nodes, the proposed method achieves better results with network sizes, in respective order, $\geq \alpha = 3 + 4\rho$, $\geq \alpha = 4 + 5\rho$, $\geq \alpha = 5 + 6\rho$, $\geq \alpha = 7 + 8\rho$. We conclude that the proposed method outperforms the referenced method in preserving node coverage, which leads to a reliable network as follows. (1) The proposed method reaches all nodes in the network with the presence of faulty nodes and preserves the network connectivity and reliability, whereas the referenced method does not reach all non-faulty nodes, i.e., disconnected nodes. (2) The proposed method outperforms the referenced method with the presence of one faulty node in all network sizes. (3) It is obvious that the more faulty nodes there are in the referenced method, the more disconnected nodes we have. As a consequence, the proposed method outperforms the referenced method with the presence of faulty nodes for large network sizes.
| $\alpha$                     | $1 + 2\rho$ (%) | $2 + 3\rho$ (%) | $3 + 4\rho$ (%) | $4 + 5\rho$ (%) | $5 + 6\rho$ (%) | $6 + 7\rho$ (%) | $7 + 8\rho$ (%) | $8 + 9\rho$ (%) | $9 + 10\rho$ (%) |
|----------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Broadcasting [36] (1 Faulty)| 85.71           | 91.22           | 93.69           | 95.08           | 95.57           | 96.58           | 97.04           | 97.38           | 97.08           |
| Proposed method (1 Faulty)  | 85.71           | 94.73           | 97.29           | 98.36           | 98.90           | 99.21           | 99.40           | 99.53           | 99.63           |
| Proposed method (2 Faulty)  | 71.42           | 89.47           | 94.59           | 96.72           | 97.80           | 98.42           | 98.81           | 99.07           | 99.26           |
| Proposed method (3 Faulty)  | 57.14           | 84.21           | 91.89           | 95.08           | 96.70           | 97.63           | 98.22           | 98.61           | 98.89           |
| Proposed method (4 Faulty)  | 42.85           | 78.94           | 89.18           | 93.44           | 95.60           | 96.85           | 97.63           | 98.15           | 98.52           |
| Proposed method (5 Faulty)  | 28.57           | 73.68           | 86.48           | 91.80           | 94.50           | 96.06           | 97.04           | 97.96           | 98.15           |
6 Conclusion

In this paper, we have presented construction techniques of edge-disjoint node-independent spanning trees (EDNIST) and node-independent spanning trees (IST) in Eisenstein–Jacobi networks. Because of the network symmetry, in EDNIST, the first tree is constructed and then it is rotated twice to obtain the second and third disjoint trees. Whereas in IST, the first tree is constructed and then it is rotated five times to get the second, third, fourth, fifth, and sixth independent spanning trees. We have shown that the depth of EDNIST is $2k + 2$ and the depth of IST is $2k + 1$. Both constructions take time complexity $O(n)$. In addition, we have presented a unified routing algorithm for both EDNISTs and ISTs with time complexity $O(1)$ and communication complexity $O(|P|)$. The proposed construction method can be easily deployed in parallel since the parent and child nodes are given for each node in the network.

The simulation presented in Sect. 5 supports the Lemmas and Theorems proved in this paper. The results show the average maximum number of steps required to construct all trees is $\leq 2k + 1$ with the presence of all possible faulty nodes. Further, the maximum of all maximum number of steps to construct all trees using all ports simultaneously is bounded to the upper bound. In addition, the proposed method outperforms the broadcasting algorithm in [36] in terms of node coverage in the network with the presence of faulty nodes.

As for future research, the proposed construction considers the same root node for all spanning trees, which is limited in case of the root node being faulty. This can be solved by having a completely independent spanning trees. The problems of finding completely independent spanning trees in Gaussian and EJ networks are our next investigation. Further, the problems of finding ISTs and EDNISTs in higher dimensional Gaussian and Eisenstein–Jacobi networks proposed in [37–39] are still open.

References

1. Itai A, Rodeh M (1988) The multi-tree approach to reliability in distributed networks. Inf Comput 79(1):43–59
2. Krishnamoorthy M, Krishnamurthy B (1987) Fault diameter of interconnection networks. Comput Math Appl 13(5):577–582
3. Rescigno AA (2001) Vertex-disjoint spanning trees of the star network with applications to fault-tolerance and security. Inf Sci 137(1):259–276
4. Yang JS, Chan HC, Chang JM (2011) Broadcasting secure messages via optimal independent spanning trees in folded hypercubes. Discret Appl Math 159(12):1254–1263
5. Darties B, Gastineau N, Togni O (2018) Almost disjoint spanning trees: relaxing the conditions for completely independent spanning trees. Discret Appl Math 236:124–136
6. Hasunuma T, Morisaka C (2012) Completely independent spanning trees in torus networks. Networks 60(1):59–69
7. Tang SM, Yang JS, Wang YL, Chang JM (2010) Independent spanning trees on multidimensional torus networks. IEEE Trans Comput 59(1):93–102
8. Ku SC, Wang BF, Hung TK (2003) Constructing edge-disjoint spanning trees in product networks. IEEE Trans Parallel Distrib Syst 14(3):213–221
9. Yang JS, Chang JM (2014) Optimal independent spanning trees on cartesian product of hybrid graphs. Comput J 57(1):93–99
10. Tang SM, Wang YL, Leu YH (2004) Optimal independent spanning trees on hypercubes. J Inf Sci Eng 20(1):143–156
11. Yang JS, Wu MR, Chang JM, Chang YH (2015) A fully parallelized scheme of constructing independent spanning trees on Möbius cubes. J Supercomput 71(3):952–965
12. Chang YH, Yang JS, Chang JM, Wang YL (2015) A fast parallel algorithm for constructing independent spanning trees on parity cubes. Appl Math Comput 268:489–495
13. Cheng B, Fan J, Jia X (2015) Dimensional-permutation-based independent spanning trees in bijective connection networks. IEEE Trans Parallel Distrib Syst 26(1):45–53
14. Wang Y, Fan J, Zhou G, Jia X (2012) Independent spanning trees on twisted cubes. J Parallel Distrib Comput 72(1):58–69
15. Chang JM, Yang TJ, Yang JS (2017) A parallel algorithm for constructing independent spanning trees in twisted cubes. Discret Appl Math 219:74–82
16. Cheng DW, Yao KH, Hsieh SY (2021) Constructing independent spanning trees on generalised recursive circulant graphs. IEEE Access 9:74028–74037
17. Yang JS, Li XY, Peng SL, Chang JM (2022) Parallel construction of multiple independent spanning trees on highly scalable datacenter networks. Appl Math Comput 413:126,617
18. Chen G, Cheng B, Wang D (2020) Constructing completely independent spanning trees in data center network based on augmented cube. IEEE Trans Parallel Distrib Syst 32(3):665–673
19. Kao SS, Klasing R, Hung LJ, Hsieh SY (2021) A parallel algorithm for constructing multiple independent spanning trees in bubble-sort networks. In: International Conference on Algorithmic Applications in Management, pp. 252–264. Springer
20. Cheng B, Wang D, Fan J (2017) Constructing completely independent spanning trees in crossed cubes. Discret Appl Math 219:100–109
21. Lin JC, Yang JS, Hsu CC, Chang JM (2010) Independent spanning trees vs. edge-disjoint spanning trees in locally twisted cubes. Inf Process Lett 110(10):414–419
22. Yang JS, Chang JM, Chan HC (2009) Independent spanning trees on hypercubes. In: Proceedings of the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks, ISPAN '09, pp. 601–605. IEEE Computer Society, Washington, DC, USA. https://doi.org/10.1109/ISPAN.2009.55
23. Yang JS, Chang JM, Pai KJ, Chan HC (2015) Parallel construction of independent spanning trees on enhanced hypercubes. IEEE Trans Parallel Distrib Syst 26(11):3090–3098
24. Cheng DW, Chan CT, Hsieh SY (2019) Constructing independent spanning trees on pancake networks. IEEE Access 8:3427–3433
25. Lin CF, Huang JF, Hsieh SY (2020) Constructing independent spanning trees on transposition networks. IEEE Access 8(147):122–147
26. AlBdaiwi B, Hussain Z, Cerny A, Aldred R (2016) Edge-disjoint node-independent spanning trees in dense gaussian networks. J Supercomput. https://doi.org/10.1007/s11227-016-1768-x
27. Hussain Z, AlBdaiwi B, AboElfotoh H (2017) Parallel construction of edge-disjoint node-independent spanning trees in dense gaussian networks. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), pp. 117–122. The Steering Committee of The World Congress in Computer Science, Computer
28. Hussain Z, AlBdaiwi B, Cerny A (2017) Node-independent spanning trees in gaussian networks. J Parallel Distrib Comput 109:324–332
29. Flahive M, Bose B (2010) The topology of Gaussian and Eisenstein–Jacobi interconnection networks. IEEE Trans Parallel Distrib Syst 21(8):1132–1142
30. Martínez C, Stafford E, Beivide R, Gabidulin EM (2008) Modeling hexagonal constellations with Eisenstein–Jacobi graphs. Probl Inf Transm 44:41–11
31. Huber K (1994) Codes over Eisenstein–Jacobi integers. Contemp Math 168:165–165
32. Chen MS, Shin K, Kandlur D (1990) Addressing, routing, and broadcasting in hexagonal mesh multiprocessors. IEEE Trans Comput 39(1):10–18
33. Dolter J, Ramanathan P, Shin K (1991) Performance analysis of virtual cut-through switching in harts: a hexagonal mesh multicomputer. IEEE Trans Comput 40(6):669–680. https://doi.org/10.1109/12.90246
34. Hussain ZA, Bose B, Al-Dhelaan A (2015) Edge disjoint Hamiltonian cycles in Eisenstein–Jacobi networks. J Parallel Distrib Comput 86:62–70
35. Networkx: Network analysis in python. https://networkx.org/. Accessed: 2021-01-08
36. Albader B, Bose B, Flahive M (2011) Efficient communication algorithms in hexagonal mesh interconnection networks. IEEE Trans Parallel Distrib Syst 23(1):69–77
37. Bose B, Shamaei A, Flahive M (2015) Higher dimensional gaussian networks. IEEE Trans Parallel Distrib Syst 27(9):2628–2638
38. Hussain Z, Shamaei A (2017) Higher dimensional Eisenstein–Jacobi networks. J Parallel Distrib Comput 102:91–102
39. Shamaei A (2015) Interconnection networks based on gaussian and Eisenstein–Jacobi integers

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.