IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 21 September 2023, accepted 13 October 2023, date of publication 30 October 2023, date of current version 2 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3326853

THEORY

Full Binary Tree-Based Fixed Degree Graph Design and Parallel Algorithm

BO OK SEONG, JIN-HYEOK JANG, AND HYEONG OK LEE
Department of Computer Education, National University of Sunchon, Suncheon-si, Chonnam 57922, Republic of Korea

Corresponding author: Hyeongok Lee (oklee@scnu.ac.kr)

This work was supported in part by the Government (Ministry of Science and ICT), and in part by the National Research Foundation of Korea under Grant 2020R1A2C1012363.

ABSTRACT Interconnection network is a network that connects processors and is an important factor in determining the performance of a parallel processing system. One measure of interconnection network evaluation is network cost, defined as degree $*$ diameter. The interconnection networks proposed so far can be classified into mesh, hypercube, and star graph types based on the number of nodes. The interconnection network $Tree - baseGraph(TG_n)$ proposed in this study is a graph based on a full binary tree with a fixed degree of three, and the node address is expressed using n binary numbers. In this study, routing algorithms, Hamiltonian cycle, node disjoint parallel path, etc. are analyzed for $Tree - baseGraph(TG_n)$. TG_n graph has network cost $O(6n)$ with a fixed degree of three and diameter $2n - 2$. From the network cost viewpoint, TG_n has around 50% improvement results compared to existing fixed degree graphs such as mesh, torus, honeycomb mesh and shuffle-exchange permutation.

INDEX TERMS Fixed degree, full binary tree, interconnection network, network cost, parallel paths.

I. INTRODUCTION

With the recent information technology development, data has explosively increased, and the COVID-19 pandemic has led to the development of artificial intelligence (AI) technologies for big data analysis and inference and prediction. The demand for high-performance computers with powerful computing capabilities is increasing in the AI, autonomous driving, and robotics industries and high-performance computers are constantly evolving in response to new demands and requirements as these application areas diversify [1]. Related research includes "Simulation-based modeling of coronavirus disease spread," "Proposal of an efficient federated learning-based deep learning optimization model in a distributed environment," and "Development of data analysis, modeling, and deep learning recognition algorithms for the electromagnetic field generation surrounding a living body" [2], [3], [4].

High-performance parallel processing refers to a technique in which multiple processors simultaneously process multiple programs or partitioned parts of a single program by dividing the workload [5], [6], [7]. Parallel processing computers can be broadly categorized into multiprocessor systems with unique memory and multicomputer systems with distributed memory. Each processor in a multicomputer system has its own local memory device, and the processors are connected by a static interconnection network. The communication between these processors is processed through message passing over the interconnection network, and the computation is data-driven [8], [9]. Supercomputers generally refer to computers that can produce, process, and utilize large amounts of data at extremely high speeds. The Top500 list, which ranks high-performance computers worldwide based on supercomputer performance criteria, is published biannually (in June and November). The Frontier supercomputer at the Oak Ridge National Laboratory (ORNL) in the United States has been ranked No. 1 for the second consecutive year in the Top500 list of the world's fastest supercomputers, released in May 2023. The 2nd and 3rd ranks are achieved by Japan's "Fugaku" and Finland's "Large Unified Modern Infrastructure (LUMI)," respectively. South Korea's supercomputers are ranked 8th in terms of performance [10].

The associate editor coordinating the review of this manuscript and approving it for publication was P. Venkata Krishna.

The interconnection network is a structure that connects processors in a high-performance parallel processing computer and is one of the determining factors of the performance of a parallel processing system [11]. This means that research on interconnection networks is needed to improve the performance of parallel processing computers. There are various metrics for evaluating interconnection networks, such as degree, diameter, symmetry, cycle, fault diameter, broadcasting, and embedding.

In this study, $Tree-baseGraph(TG_n)$ with a fixed degree of three is proposed using a full binary tree. A routing algorithm was designed for TG_n and implemented the algorithm considering the degree, diameter, network cost, Hamiltonian cycle, and node-disjoint parallel path among the network metrics.

II. INTERCONNECTION NETWORK AND FIXED DEGREE GRAPH

In a multiprocessor system, the connection network that supports communication between processors is called a multiprocessor connection network [12]. The interconnection network of parallel computers is represented as an undirected graph $G = (V(G), E(G))$. $V(G)$ refers to the set of nodes that constitute the graph G, and $E(G)$ refers to the set of edges [13]. The network evaluation metrics for the interconnection network include the degree, diameter, network cost, connectivity, symmetry, and Hamiltonian cycle.

The degree is the maximum value of edges adjacent to any node in the interconnection network G. The diameter is the maximum value of the shortest path between any two nodes. The degree refers to the hardware cost of building a system using Graph G, and the diameter refers to the software processing cost for data transmission in the system. The degree and diameter of a graph are inversely related. In general, as the degree increases, the diameter decreases, and vice versa. The network cost is the value derived by multiplying the degree by the diameter. The connectivity is the minimum number of nodes (or edges) that need to be removed to separate the remaining nodes into two or more disjoint subgraphs. In terms of symmetry, it is deemed symmetric when auto morphism that corresponds to any two nodes or edges in the graph exists [14], [15]. In a connected graph G, a Hamiltonian cycle is a cycle that includes all vertices. If an interconnection network has a Hamiltonian path or a Hamiltonian cycle, it can be easily implemented as a ring or linear array, which can be used as a pipeline that is useful for parallel processing [16].

The proposed interconnection networks thus far can be categorized based on the number of nodes as follows: Mesh category with $k \times n$ nodes, hypercube category with 2^n nodes, star graph category with $n!$ nodes, etc. [17]. Hypercube and star graph categories have the characteristic that the number of edges increases as the number of nodes increases.

The node addresses in an n-dimensional hypercube are represented by n-bit binary numbers, and any two nodes with addresses that differ by exactly one bit are connected. The hypercube graph is node and edge symmetric and has a bipartite graph structure. While it has the advantage of being able to efficiently embed interconnection network structures such as rings, trees, and meshes, it has the disadvantage that the node degree increases proportionally as the dimension increases [18]. To improve this disadvantage, graphs such as folded hyper cubes, multiply-twisted-cube, gray cubes, and recursive circulants have been proposed by changing the relationship between nodes and edges.

The star graph is node and edge symmetric and has excellent scalability through a recursive structure. It has the advantage of having a smaller number of degrees and diameters compared to the hypercube. However, its disadvantages include the difficulty with being embedded with other interconnection network structures and the number of nodes increasing rapidly with the increase in dimensions [26], [27], [28], [29], [30]. To address the disadvantage of the star graph, several alternatives have been proposed, such as an alternating group [31], star-connected cycles graph [29], (n, k)-star graph [29], bubble-sort star graph [27], transposition graph [15], [32], and macro-star graph [33].

For the increase in the number of nodes in interconnection networks, various graphs with a fixed degree have been proposed, such as mesh, torus, shuffle exchange [19], de Bruijn [20], butterfly [21], cube-connected-cycle [22], diagonal mesh, honeycomb mesh, Shuffle-Exchange Permutation (SEP) [31], new SEP (NSEP) [37], and others [15], [23], [24], [25], [26].

A mesh structure is a planar graph with a grid structure and has been widely utilized and commercialized in the form of various systems [34], [35]. Improved structures for mesh with reduced diameter include torus, diagonal mesh, honeycomb mesh, etc. [35], [36]. The SEP graph has the advantage of being able to easily simulate algorithms of graphs based on permutation groups, similar to Cayley graphs. In recent years, NESP with a degree of four has been proposed to improve the network cost of the SEP graph [37].

The m-dimensional mesh $M_m(N)$ consists of N^m nodes and $mN^m - mN^{m-1}$ edges. Low-dimensional meshes are easy to design and very useful from an algorithmic perspective, so they are often used as the interconnection network of parallel processing computers [34]. As the dimension of the mesh increases, the diameter becomes smaller, and the bipartite width becomes larger, allowing for faster execution of various parallel algorithms. However, this comes at the cost of increased expenses [17], [21]. The torus graph is a network that adds wraparound edges to the mesh structure, with a degree of four and a diameter of $\sqrt{N}$ [38]. The honeycomb mesh can be made in three ways using hexagons [35], [36]. Honeycomb hexagonal mesh (HHM), honeycomb rhombic mesh (HRoM), and honeycomb rectangular mesh (HReM) can be made by connecting hexagons in different ways. Each of these meshes has a degree of three, and adding wraparound edges to them creates a honeycomb torus. HHM is simply called honeycomb mesh (HM). HM HM_t has $6t^2$ nodes, $9t^2 - 3t$ edges, and a degree of three [37]. The nodes of the SEP_n graph are regular graphs represented by permutations

TABLE 1. **Comparison of fixed degree graphs.**

Interconnecti network	No. of nodes	No. of degrees	Diameter	Network cos
Mesh	$k \times n$	4	$2\sqrt{n}$	$O(8\sqrt{n})$
Honeycomb mesh	$6t^2$	3	$1.63\sqrt{n}$	$O(4.9\sqrt{n})$
Torus	$k \times n$	4	$\sqrt{n}$	$O(4\sqrt{n})$
SEP	$n!$	3	$\frac{1}{8}(9n^2 - 22n + 24)$	$O(\frac{27}{8}n^2)$
NSEP	$n!$	4	$\frac{2}{3}n^2 - \frac{3}{2}n + 1$	$O(\frac{8}{3}n^2)$

of the set $\{1, 2, \cdots, n\}$ and having a degree of three. The SEP_n graph can be easily simulated in graphs, which are based on permutation groups such as Cayley graphs and algorithms can be efficiently executed on the new graph with minimal changes [31]. Interconnection network $NSEP_n$ has $n!$ nodes, a degree of four, a diameter of $\frac{2}{3}n^2 - \frac{3}{2}n + 1$, and network cost $Q(n^2)$. While the degree increased by one in the interconnection network $NSEP_n$ compared to that of SEP_n, its improvement outcome includes more than 40% for diameter and more than 20% for network cost [39].

III. TREE-BASED GRAPH DEFINITION AND PROPERTIES

Tree − baseGraph(TG_n) refers to a graph with a degree of three using a full binary tree. The node address for TG_n is represented by n binary numbers. In this study, the node S's address in TG_n will be represented as an integer corresponding to the binary number ($0 \leq \mathrm{S} \leq 2^{\mathrm{n}} - 1$).

Definition 1: The n-dimensional TG_n graph is represented by n binary numbers ($n \geq 2$). The node addresses of TG_n are conveniently represented as integers, ranging from $\{0, 1, 2, \cdots, \cdots, 2^{\mathrm{n}}-2, 2^{\mathrm{n}}-1\}$, and the total number of nodes is 2^{n}.

Definition 2: The nodes that make up the node sets TR, TI, and TT in TG_n are defined as follows:

Note set TR: At index $I = 0$, the node addresses $\{0\}$

Note set TI: At index $1 \leq I \leq n-1$, the node addresses $\{1, 2, 3, \ldots, 2^{n-1} - 2, 2^{n-1} - 1\}$

Note set TT: At index $I = n$, the node addresses $\{2^{n-1}, 2^{n-1} + 1, 2^{n-1} + 2, \ldots, 2^n - 2, 2^n - 1\}$

Definition 3: The edges constituting the TG_n graph are defined as ER, EI, and ET according to the node addresses: $\{0\}$ for ER, $\{1 - 2^{n-1} - 1\}$ for EI, and $\{2^{n-1}–2^n - 1\}$, respectively.

When the node address is denoted as $S (0 \leq S \leq 2^n - 1)$, the adjacent nodes to node S are as follows:

Edge ER: Connecting the node TR $=\{0\}$ and the node $\{1, 2^{n-1}, 2^n - 1\}$

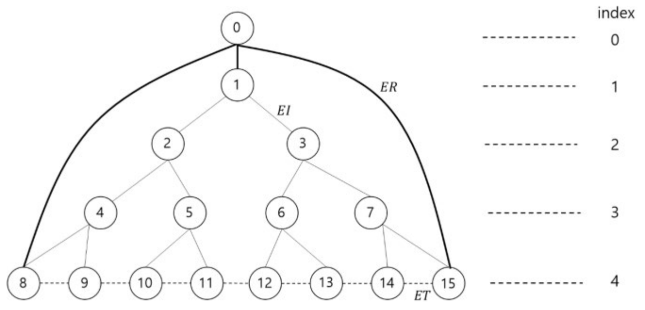

FIGURE 1. TG_4 **graph.**

Edge EI: Connecting the node TI $=\{1, 2, 3, \ldots, 2^{n-1} - 2, 2^{n-1} - 1\}$ and the node $\{2S, 2S + 1, \lfloor \frac{S}{2} \rfloor\}$

Edge ET: Connecting the node TT $= \{2^{n-1}, 2^{n-1} + 1, 2^{n-1} + 2, \ldots, 2^n - 2, 2^n - 1\}$ and the node $\{S - 1, S + 1, \lfloor \frac{S}{2} \rfloor\}$

Note that node addresses 2^{n-1} and $2^n - 1$ of the node set TT have only one adjacent node, which is connected through index I. Thus, the node addresses adjacent to the node address 2^{n-1} are $\{0, S + 1, \lfloor \frac{S}{2} \rfloor\}$, and the node addresses adjacent to the node address $2^n - 1$ are $\{S - 1, 0, \lfloor \frac{S}{2} \rfloor\}$.

Definition 4: The index I, which represents the location of the node in the TG_n graph, means the distance between any node connected by the edge EI from the node TR ($0 \leq I \leq n$). Index I can be divided into three cases ($I = 0$, $1 \leq I \leq n-1$, $I = n$) and represented by node sets TR, TI, and TT depending on each case.

Figure 1 shows the node addresses and index I in the TG_4 graph ($0 \leq I \leq 4$). The node addresses are represented by $\{0, 1, 2, \cdots, 15\}$ according to Definition 1 and the number of nodes is 16. According to Definition 2, TR $=0$, TI $= \{1, 2, 3, 4, 5, 6, 7\}$, and TT $= \{8, 9, 10, 11, 12, 13, 14, 15\}$. In Figure 1, edges ER, EI, and ET are represented by the bold, solid, and dotted lines, respectively. At index $I = 4$ in TG_4, the connected node set of node addresses 8 and 15 in node set TT is as follows by Definition 3. The node addresses adjacent to node 8 are $\{0, 4, 9\}$ and the node addresses adjacent to node 15 are $\{0, 7, 14\}$.

Corollary 1: Graph TG_n has a full binary tree with level n-1 as a subgraph.

Proof: Removing the edges ER that connect the node set $\{1, 2^{n-1}, 2^n - 1\}$ adjacent to node set TR (node address 0) in the TG_n and the $2^{n-1} - 1$ edges ET that connect the node at index n results in a full binary tree structure with level n-1. □

Property 1: Graph TG_n is a connected graph.

Proof) According to Corollary 1, TG_n has a full binary tree as a subgraph. The added node is the root node TR whose node address is 0, which increases the number of nodes by 1 compared to that in the full binary tree structure. TR is connected to node $\{1, 2^{n-1}, 2^n - 1\}$ through edge ER according to Definition 3. Edge EI is the same as the edge of the full binary tree and edge ET is an edge that connects nodes at index n. Thus, graph TG_n is a connected graph. □

Since TG_n is a graph that represents nodes with n bits based on the full binary tree, it has 2^n nodes.

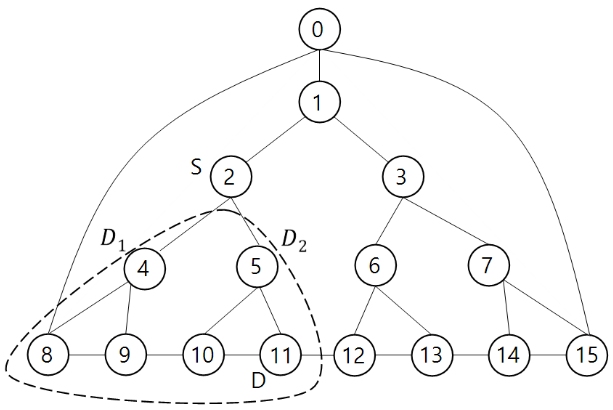

FIGURE 2. Direct and successor relationships of nodes S, D, D1, and D2 in TG_4.

Property 2: The number of nodes in TG_n is 2^n.

Proof: Since TG_n has a full binary tree as a subgraph, its number of nodes is $2^n - 1$. Node TR $= \{0\}$ was added to index 0 in the full binary tree of TG_n. Thus, the number of nodes in TG_n is 2^n, which is one more than the number of nodes in the full binary tree. □

Property 3: In the node address S in TG_n, if $S = 0$, index I is 0 and if $S \geq 1$, index I is $\lfloor log_2 S + 1 \rfloor$ $(0 \leq S \leq 2^n - 1)$.

Routing is a path to transmit messages between two nodes in the network [37]. Assuming that the start node is S and the destination node is D, symbols used in the algorithm and routing algorithms of TG_n are presented.

The routing algorithm of TG_n is performed by examining the direct or successor relationship between any two nodes.

Definition 5: The direct and successor nodes in TG_n are represented as follows: Let us assume any node S and let D_1, D_2 be any two of its three adjacent nodes whose index value is larger than S's. Similarly, there is a node whose index value is greater than that of D_1, D_2 among the adjacent nodes of nodes D_1, D_2. Let the set of these nodes be the successor nodes of S. As shown in Figure 2, if any node D belongs to the successor node set of S, S is called the direct node of D, and D is called the successor node of S.

When $n = 4$, node 2 is a direct node of 11. The nodes whose index value is greater than that of 2 among three nodes adjacent to node 2 are $\{4, 5\}$. The nodes whose index value is greater among nodes adjacent to node $\{4, 5\}$ are $\{8, 9, 10, 11\}$. Thus, the successor node set of node 2 is $\{4, 5, 8, 9, 10, 11\}$. As a result, node 11 is a successor node of node 2 and node 2 is a direct node of node 11.

Theorem 1: When node S in TG_n is a direct node of D, it satisfies the following equation. Node S's index is a and node D's index is b (b > a).

$$S \times 2^{b-a} \leq D \leq S \times 2^{b-a} + 2^{b-a} - 1$$

Proof: Graph TG_n satisfies the following case according to Corollary 1.

Case 1: $S \times 2^{b-a} \leq D$

$b - a$ refers to the difference in the indexes of two nodes and $S \times 2^{b-a}$ is the smallest value of the nodes in the index to which the successor node D of S belongs. Thus, it satisfies $S \times 2^{b-a} \leq D$.

Case 2: $D \leq S \times 2^{b-a} + 2^{b-a} - 1$

2^{b-a} represents the number of nodes in the index where D belongs among the successor nodes of S. We verified in Case 1 that D's minimum value was $S \times 2^{b-a}$. The number of node addresses of D is $S \times 2^{b-a}$ to 2^{b-a}. Thus, the maximum value of D is $S \times 2^{b-a} + 2^{b-a} - 1$. Thus, it satisfies $D \leq S \times 2^{b-a} + 2^{b-a} - 1$.

Accordingly, it satisfies $S \times 2^{b-a} \leq D \leq S \times 2^{b-a} + 2^{b-a} - 1$. □

Definition 6: Simple routing (U, V) is defined as follows:

Simple routing (U, V) is a path from a starting node U to a destination node V. First, verify whether U and V have a direct or successor relationship using the index values of the nodes. Then, by applying the equation in Theorem 1, decrease the index or select and move to the successor node where the destination node belongs to.

Case 1. When node V is a direct node of U: Decrease the index and move to the destination node.

Case 2. When node V is a successor node of U: Select a node where the successor node belongs to and move to the destination node.

Case 3. In the case where it is not a direct or successor relationship: Decrease the index until U becomes a direct node of V and select the node by applying the equation.

```
Simple Routing(S, D)

$a = \lfloor log_2 S + 1 \rfloor$, b = $\lfloor log_2 D + 1 \rfloor$

if($S = 0$)

    $S = 1$;

while(!($S \times 2^{b-a} \leq D \leq S \times 2^{b-a} + 2^{b-a} - 1$))

    $S = \lfloor \frac{S}{2} \rfloor$;

while($S \neq D$){

    if($S \times 2^{b-a} \leq D \leq S \times 2^{b-a} + 2^{b-a} - 1$)

        $S = S \times 2$;

else

        $S = S \times 2 + 1$;

}
```

According to the algorithm, examine whether or not the starting node is a direct node of the destination node. Examine the case in the graph TG_4 where the starting node is a direct node of the destination node. Let us assume that starting node

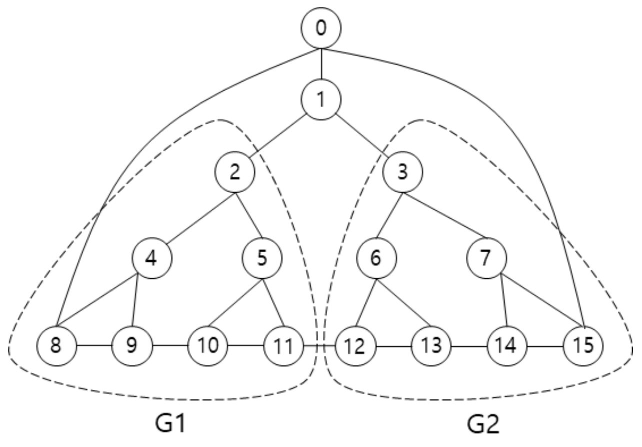

FIGURE 3. Two subgraphs $G1$, $G2$ except for nodes 0 and 1 in TG_4.

is 2 and the destination node is 10 and '$\rightarrow$' represents the routing edge. The indexes of nodes 2 and 10 are 2 and 4, and the path is as follows:

$$\text{Path}: 2 \rightarrow 5 \rightarrow 10$$

Examine when the starting node is not a direct node of the destination node. Assuming that starting node is 4 and the destination node is 7, the path is as follows:

$$\text{Route}: 4 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 7$$

Theorem 2: TG_n's diameter is $2n - 2$.

Proof: The simple routing algorithm of TG_n uses edge EM only except when the starting node is 0. In short, it uses edges like a tree. Let $G1$, $G2$ be two subgraphs consisting of all nodes except nodes 0 and 1, as shown in Figure 3. The worst case is when the starting and destination nodes belong to the TTs of $G1$, $G2$, respectively. To move from $G1$ to $G2$ using only edge EI, we need to move up to index 1. Since the index of the node set TT is n, the distance to index 1 is $n - 1$. Similarly, the distance to the destination node of the node set TT is $n - 1$, so the diameter of TG_n is $2n - 2$. □

In a connected graph G, a Hamiltonian cycle is a cycle that includes all vertices of G. If an interconnection network contains a Hamiltonian path or a Hamiltonian cycle, it can be easily implemented as a ring or linear array, which can be used as a pipeline that is useful for parallel processing [19]. If G contains a Hamiltonian cycle, even if a node or edge fails, it has a subgraph with a linear array that includes all the remaining nodes [41].

Definition 7: The starting node of the Hamiltonian cycle algorithm $H(n)$ in TG_n is 2^{n-1}. The starting node of $H(n)$ is the node with $indexI = n$, which according to Definition 2, is the smallest 2^{n-1} in the node set TT, consisting of nodes $\{2^{n-1}, 2^{n-1} + 1, 2^{n-1} + 2, \ldots, 2^n - 2, 2^n - 1\}$.

Definition 8: The Hamiltonian cycle algorithm $H(n)$ in TG_n uses the function $B(m)$, and the algorithm is as follows (m >0). Let us assume that the starting node is S and the destination node is V. $B(m)$ The function moves from S to the node with the lesser value of index I, which is connected to the edge EI, m times. Then, it moves m times to the node with the greater value of index I. The paths of the functions $B(1)$, $B(2)$ are shown in Figure 4. $B(1)$ is a path that moves once to the node with the lesser value of index I and then moves once to the node with the greater value of index I. $B(2)$ is a path that moves twice each time. In other words, V = S+1.

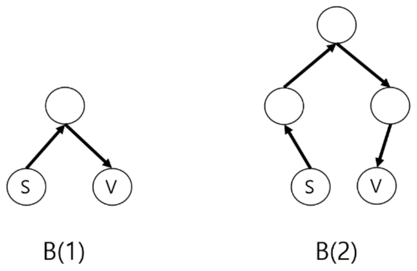

FIGURE 4. Hamiltonian cycle's functions $B(1)$, $B(2)$.

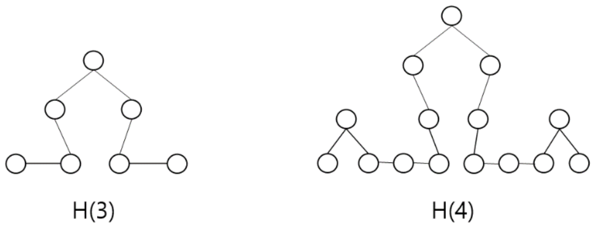

FIGURE 5. Subgraphs of the Hamiltonian cycles $H(3)$, $H(4)$ when $n = 3, 4$.

Corollary 2: The Hamiltonian cycle of TG_2 and TG_3 is defined as follows:

The algorithm of TG_2 and TG_3 is fixed. According to Definition 7, the starting node of TG_2 is 2. That is, the Hamiltonian cycle of TG_2 is as follows:

$$\text{Hamiltoniancycle}: 2 \rightarrow 1 \rightarrow 3 \rightarrow 0 \rightarrow 2$$

The starting node of TG_3 is 4, and the Hamiltonian cycle is as follows:

$$4 \rightarrow 5 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 0 \rightarrow 4$$

Figure 5 shows a partial graph of the Hamiltonian cycle of TG_3 and TG_4, with node TR, i.e., node 0, removed.

Corollary 3: When $n = 4$, the Hamiltonian cycle is defined as follows:

TG_3 and TG_4 are special cases that are recursively called when $n \geq 5$. Thus, TG_3 and TG_4 are defined as corollaries. Assuming that the starting node of TG_4 is S, $S = 8$ according to Definition 7. That is, the Hamiltonian cycle of TG_4 is as follows, which is indicated within the parenthesis: According to Definition 8, the first B(1) is $(8 \rightarrow 4 \rightarrow 9)$ and B(3) is $(5 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow 12)$, and the second B(1) is $(14 \rightarrow 7 \rightarrow 15)$.

$$\begin{aligned} &S \rightarrow B(1) \rightarrow S+2 \rightarrow S+3 \rightarrow B(3) \\ &\qquad \rightarrow S+5 \rightarrow B(1) \rightarrow 0 \rightarrow S: \\ &(8 \rightarrow 4 \rightarrow 9) \rightarrow 10 \rightarrow 11 \rightarrow (5 \rightarrow 2 \rightarrow 1 \rightarrow 3 \\ &\qquad \rightarrow 6 \rightarrow 12) \rightarrow 13 \rightarrow (14 \rightarrow 7 \rightarrow 15) \rightarrow 0 \rightarrow 8 \end{aligned}$$

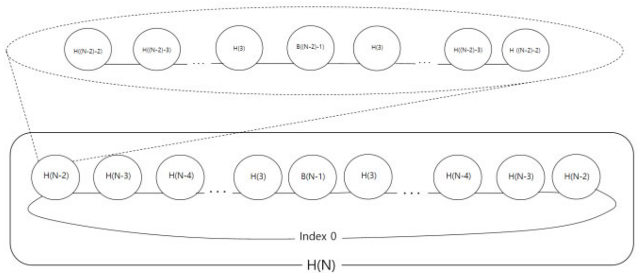

FIGURE 6. **Hamiltonian cycle $H\ (n)$ of TG_n.**

Theorem 3: The Hamiltonian cycle $H(n)$ of TG_n is defined as follows:

$H\ (3)$, $H(4)$ were defined in Corollary 3.

When $n \geq 5$, $H(n)$ calls $H(n-2)$ to $H(3)$ in sequence and passes through node 1 through the $B(n-1)$ function. After calling the functions from $H(3)$ to $H(n-2)$ in sequence, it arrives at the starting node through node TR.

Corollary 4: The Hamiltonian cycle algorithm $H(n)$ of TG_n is recursive when $n \geq 5$.

The starting node of $H(n)$ is the node address 2^{n-1} according to Definition 7. As shown in Figure 6, $H(n)$ can be largely divided into four parts as follows:

First, call the function recursively from $H(n-2)$ to $H(3)$.

Second, call $B(n-1)$ that passes through index 1.

Third, call the function from $H(3)$ to $H(n-2)$ (Symmetric to the first part).

Fourth, move to the starting node through node TR.

Property 4: The subgraph of the Hamiltonian cycle $H(n)$ of TG_n, excluding node 0, is symmetric.

TG_n is symmetric, as it recursively calls the functions around $B(n-1)$ function in accordance with Corollary 4, as can be seen in Figure 6. Thus, the subgraph of the Hamiltonian cycle $H(n)$ of TG_n, excluding node 0, is symmetric.

Hamiltonian cycle algorithm

```
B(n){
for(i = 1; i ≤ n ≤ i++)
    $S = \frac{S}{2}$;
for(i = 1; i ≤ n ≤ i++){
if(i = 1)
    $S = S \times 2 + 1$;
else
    $S = S \times 2$;
}}
H(n){
if(n = 2)
    B (1) ;
elseif(n = 3){
    $S = S + 1$;
    B(2);
    $S = S + 1$;
}
else {
 for(i = n − 2; i ≥ 3; i − −){
     H (i) ;
     $S = S + 1$;
 }
 B(1);
 for(i = 1; i ≤ 2; i + +){
     $S = S + 1$;
 }
 B(n − 1);
 for(i = 1; i ≤ 2; i + +){
     $S = S + 1$;
 }
 B(1);
 for(i = 3; i ≤ n − 2; i + +){
     $S = S + 1$;
     H(i);
} }
S = 0;
$S = 2^{n-1}$;
}
```

n of the Hamiltonian cycle $H(n)$ can be divided into odd and even numbers.

When $n \geq 5$ and n is even number $n = 6$, the Hamiltonian cycle $H(6)$ is as follows:

$H(n)$ is recursive when $n \geq 5$ and is symmetric around $B(5)$ according to Property 4. The called functions are $H\ (4)\ , H(3)$, which range from $n-2$ to $n \geq 3$. If we only look at the recursively called functions, we can see that $H\ (4)\ , H(3)$ are called in sequence, followed by $B\ (1)\ , B\ (5)\ , B\ (1), H\ (3)\ , H(4)$, which are symmetrical. The moving path is shown below and the Hamiltonian cycle is completed by passing through node TR and returning to the starting node. The moving path of $H\ (6)$ can be found in Figure 7. For convenience, after performing H(i), the TT nodes are represented as S', S'', S''' in sequence.

$$\begin{aligned} &S \rightarrow H\ (4) \rightarrow S' \rightarrow H\ (3) \rightarrow S'' \rightarrow B\ (1) \rightarrow S'' \\ &\qquad\qquad + 2 \rightarrow S'' + 3 \rightarrow B\ (5) \rightarrow \\ &S'' + 5 \rightarrow S'' + 6 \rightarrow B\ (1) \rightarrow S'' \\ &\qquad\qquad + 8 \rightarrow H\ (3) \rightarrow S''' \rightarrow H\ (4) \rightarrow 0 \rightarrow S \end{aligned}$$

When $n \geq 5$ and n is odd number $n = 7$, the Hamiltonian cycle $H(7)$ is as follows:

$H(n)$ is recursive when $n \geq 5$ and is symmetric around $B(6)$ according to Property 4. The called functions are $H\ (5)\ , H\ (4)\ , H\ (3)\ ,$ which range from $n-2$ to $n \geq 3$. If we only look at the recursively called functions, we can see that $H\ (5)\ , H\ (4)\ , H(3)$ are called in sequence, followed by $B\ (1)\ , B\ (6)\ , B\ (1), H\ (3)\ , H\ (4)\ , H\ (5)$, which are

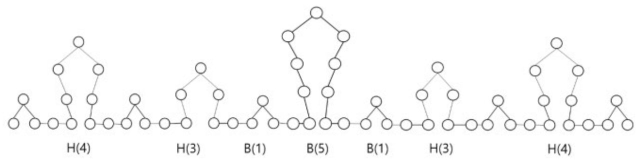

FIGURE 7. Subgraphs of the Hamiltonian cycles $H(6)$ when $n = 6$.

symmetrical. The travel path is shown below and finally completes the Hamiltonian cycle by passing through node TR and returning to the starting node.

$$S \to H(5) \to S' \to H(4) \to S'' \to H(3) \to S''' \to B(1) \to S''' + 2 \to S''' + 3 \to B(6) \to S''' + 5 \to S''' + 6 \to B(1) \to S''' + 8 \to H(3) \to S'''' \to H(4) \to S''''' \to H(5) \to 0 \to S$$

When there are multiple node-disjoint paths between any two nodes, it has the advantage of increasing speed when transferring large amounts of data between two nodes, as well as allowing different paths to be selected in the event of a failure [19], [21]. Assuming that the starting node of TG_n is A and the destination node is B, three node-disjoint parallel path algorithms, Tree(), Inside(), and Outside(), are proposed in the routing path as follows:

```
  Parallel path algorithm
Starting node A, Destination node B
$a = log_2(A) + 1$
$b = log_2(B) + 1$
Check(A, a)
{
 $while(a \neq b)\{$
   $A = A * 2;$
   $a = log_2(A) + 1;$
      }
      $if\ (A < B)$ return true;
      $else$ return false;
}
Simple Routing(A, a)
```

```
Inside(A, a)
{
  $if(A = 0)\{$
    $A = 2^{n-1};$
    $for(i = A; i < B * 2^{n-b}; i++)A = A + 1;$
    $for(i = N; i > b; i--)A = \lfloor \frac{A}{2} \rfloor;$
    $\}else\{$
     $if(!(A * 2^{b-a} \leq B \leq A * 2^{b-a} + 2^{b-a} - 1))\{$
      $if$(Check(A, a)) {
        $for(i = a; i < n; i++)A = A * 2 + 1;$
      $for(i = A; i < B * 2^{n-b}; i++)A = A + 1;$
      $for(i = n; i > b; i--)A = \lfloor \frac{A}{2} \rfloor;$
    $\}else\{$
      $for(i = a; i < n; i++)A = A * 2;$
     $for(i = A; i > B*2^{n-b}+2^{n-b}-1; i--)A = A-1;$
     $for(i = n; i > b; i--)A = \lfloor \frac{A}{2} \rfloor;$
      }
    $\}else\{$
     $if(A * 2^{b-a} \leq B \leq A * 2^{b-a} + 2^{b-a-1} - 1)\{$
       $A = A * 2 + 1;$
        $for(i = a; i < n; i++)A = A * 2;$
        $for(i = A; i > B * 2^{n-b} + 2^{n-b} - 1; i--)A =$
$A - 1;$
      $for(i = n; i > b; i--)A = \lfloor \frac{A}{2} \rfloor;$
    $\}else\{$
     $A = A * 2;$
        $for(i = a; i < n; i++)A = A * 2 + 1;$
     $for(i = A; i < B * 2^{n-b}; i++)A = A + 1;$
     $for(i = n; i > b; i--)A = \lfloor \frac{A}{2} \rfloor;$
 } } } }
```

```
Outside(A, a)
{
   $if(A = 0)\{$
   $A = 2^n - 1;$
   $for(i = A; i < B*2^{n-b}+2^{n-b}-1; i--)A = A-1;$
   $for(i = n; i > b; i--)A = \lfloor \frac{A}{2} \rfloor;$
   $\}else\{$
     $if(!(A * 2^{b-a} \leq B \leq A * 2^{b-a} + 2^{b-a} - 1))\{$
      $if$(Check(A, a)) {
      $for(i = a; i < n; i++)A = A * 2;$
      $for(i = A; i > 2^{n-1}; i++)A = A - 1;$
      $A = 0;$
      $A = 2^n - 1;$
      $for(i = A; i > B * 2^{n-b} + 2^{n-b} - 1; i--)A =$
$A - 1;$
      $for(i = n; i > b; i--)A = \lfloor \frac{A}{2} \rfloor;$
   $\}else\{$
     $for(i = a; i < n; i++)A = A * 2 + 1;$
   $for(i = A; i < 2^n - 1; i++)A = A + 1;$
     $A = 0;$
   $A = 2^{n-1};$
     $for(i = A; i < B * 2^{n-b}; i++)A = A + 1;$
   $for(i = n; i > b; i--)A = \lfloor \frac{A}{2} \rfloor;$        }
   $\}else\{$
     $if(A * 2^{b-a} \leq B \leq A * 2^{b-a} + 2^{b-a-1} - 1)\{$
     $for(i = a; i > 0; i--)A = \lfloor \frac{A}{2} \rfloor;$
     $A = 2^{n-1};$
     $for(i = A; i > B * 2^{n-b}; i++)A = A + 1;$
```

```
      $for(i = n; i > b; i--)A = \lfloor \frac{A}{2} \rfloor;$
    $\}else\{$
      $for(i = a; i > 0; i--)A = \lfloor \frac{A}{2} \rfloor;$
      $A = 2^n - 1;$
    $for(i = A; i > B*2^{n-b} + 2^{n-b} - 1; i--)A = A - 1;$
    $for(i = n; i > b; i--)A = \lfloor \frac{A}{2} \rfloor;$
  } } } }
```

Lemma 1: The Inside () algorithm is as follows:

Let A be an arbitrary starting node and B an arbitrary destination node. The Inside() algorithm can be broadly categorized into two cases: when $A = 0$ and when $A \neq 0$.

Case 1: When $A = 0$,

1-1 $A = 2^{n-1}$

1-2 There exists a successor of B $TT_i, TT_j (2^{n-1} \leq TT_i, TT_j \leq 2^n - 1, i < j)$. Iterate the following until it satisfies $A = TT_i$.

$$A = A + 1$$

1-3 Iterate the following until reaching B.

$$A = \left\lfloor \frac{A}{2} \right\rfloor$$

Case 2: When $A \neq 0$,

2-1 Iterate the following according to the condition until it satisfies $A \in TT$.

$$A = \left\lfloor \frac{A}{2} \right\rfloor or A = \left\lfloor \frac{A}{2} \right\rfloor + 1$$

2-2 When $A < TT_i$, iterate the following until it satisfies $A = TT_i$.

$$A = A + 1$$

2-3 When $A > TT_j$, iterate the following until it satisfies $A = TT_j$.

$$A = A - 1$$

2-4 When $A = TT_i$, iterate the following according to the condition until reaching B.

$$A = \left\lfloor \frac{A}{2} \right\rfloor or A = \left\lfloor \frac{A}{2} \right\rfloor + 1$$

Lemma 2: The Outside () algorithm is as follows:

The Outside () algorithm is categorized into three cases. First, when the starting node $A = 0$. Second, when A is neither a direct nor a successor node of B. Third, when A is a direct or successor node of B.

Case 1: When $A = 0$,

1-1 $A = 2^n - 1$

1-2 There exists a successor of B $TT_i, TT_j (2^{n-1} \leq TT_i, TT_j \leq 2^n - 1, i < j)$. Iterate the following until it satisfies $A = TT_j$.

$$A = A - 1$$

1-3 Iterate the following until reaching B.

$$A = \left\lfloor \frac{A}{2} \right\rfloor$$

Case 2: $A \neq B$'s direct node

The index of A is a and the index of B is b.

2-1 Compare the size of A and B using the Check () function.

2-2 Iterate the following according to the condition until it satisfies $A \in TT$.

$$A < B : A = A * 2, \quad A > B : A = A * 2 + 1$$

2-3 Iterate the following until it satisfies $A \in \{2^{n-1}, 2^n - 1\}$.

$$A < B : A = A - 1, \quad A > B : A = A + 1$$

2-4 $A = 0$

2-5 $A = 2^n - 1$

2-6 Iterate the following according to the condition until it satisfies $A = TT_j$.

$$A < B : A = A - 1, \quad A > B : A = A + 1$$

2-7 Iterate the following until reaching B.

$$A = \left\lfloor \frac{A}{2} \right\rfloor$$

Case 3: $A = B$'s direct node

3-1 Iterate the following until it satisfies $A \in TT$.

$$A = \left\lfloor \frac{A}{2} \right\rfloor$$

3-2 Move closer to B according to the condition.

$$A = 2^{n-1} or A = 2^n - 1$$

3-3 Iterate the following according to the condition.

$$A = A + 1 or A = A - 1$$

3-4 Iterate the following until reaching B.

$$A = \left\lfloor \frac{A}{2} \right\rfloor$$

Theorem 4: The routing paths of algorithms Tree (), Inside (), and Outside () are node-disjoint parallel paths.

Proof) That they are parallel algorithms is demonstrated by the fact that the paths in all three algorithms have disjoint nodes. Assuming that starting node is A and the destination node is B, let us assume that arbitrary nodes of TT are TT_i, TT_j, TT_k, TT_l ($2^{n-1} \leq i, j, k, l \leq 2^n - 1$, i≠j≠k≠l)

Case 1: When A =0,

Inside() and Outside() algorithms start from node TR and each pass through nodes 2^{n-1} and $2^n - 1$, then go through the successor nodes of B, TT_i and TT_j, and finally reach B through the direct node. The path can be found in Figure 8.

Case 2: When node A is the direct node of B,

Inside() bypasses the direction in which B is located and reaches the node TT_j, and then reaches B through the node

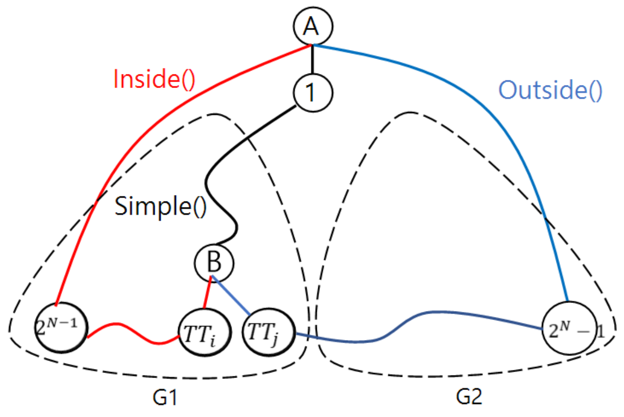

FIGURE 8. Case 1 of TG_n's parallel path.

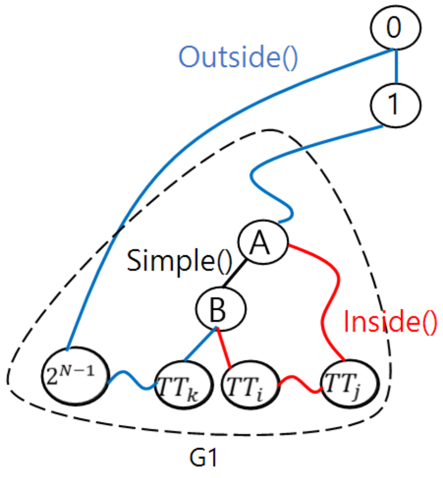

FIGURE 9. Case 2 of TG_n's parallel path.

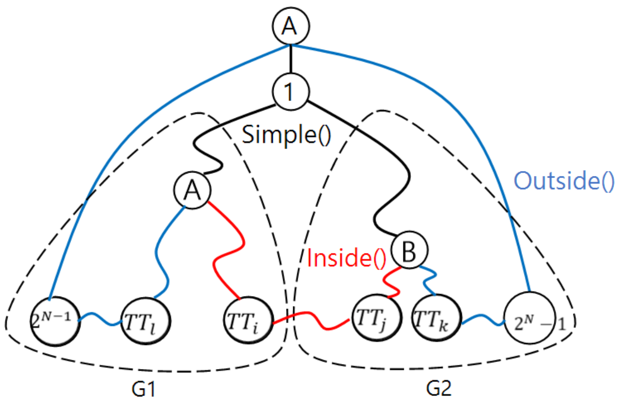

FIGURE 10. Case 3 of TG_n's parallel path.

TT_i. Outside() moves from node A to TR, then searches for the node closest to B among 2^{n-1} and $2^n - 1$, and reaches B through node TT_k. The path can be found in Figure 9.

Case 3: When the above two conditions are not satisfied

TABLE 2. Fixed degree graphs and TG_n.

Interconnectio network	No. of nodes	No. of degrees	Diameter	Network cost
Mesh	$k \times n$	4	$2\sqrt{n}$	$O(8\sqrt{n})$
Honeycomb mesh	$6t^2$	3	$1.63\sqrt{n}$	$O(4.9\sqrt{n})$
Torus	$k \times n$	4	$\sqrt{n}$	$O(4\sqrt{n})$
SEP_n	$n!$	3	$\frac{1}{8}(9n^2 - 22n + 24)$	$O(\frac{27}{8}n^2)$
$NSEP_n$	$n!$	4	$\frac{2}{3}n^2 - \frac{3}{2}n + 1$	$O(\frac{8}{3}n^2)$
TG_n	2^n	3	$2n-2$	$O(6n)$

TABLE 3. Comparison of network cost when the number of nodes is the.

No. of nodes	Network cost					
	Mesh	Honeycomb	Torus	SEP	NSEP	TG
120	23.53	10.00	13.20	52	40.67	42
720	41.50	16.00	20.70	81	64.00	60
5040	67.40	26.30	33.70	117	92.67	78
40320	113.40	44.20	56.70	159	126.67	90
362880	196.45	120.08	98.22	208	166.00	110
3628800	349.17	213.43	174.59	264	210.67	132
39916800	635.89	388.69	317.94	327	260.67	150

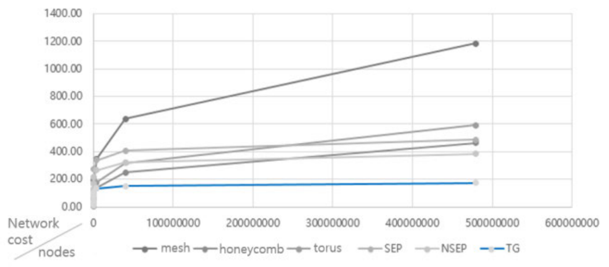

FIGURE 11. Comparison of the network costs of fixed degree graphs when the number of nodes is the same.

Inside() searches the nodes close to B and moves to node TT_i. Then, it reaches B through TT_j, the closest successor node of B. Outside(), on the other hand, moves in the direction away from B and reaches node TT_l. Then, it arrives at either 2^{n-1} or $2^n - 1$. After that, it passes through TR and moves towards 2^{n-1} or $2^n - 1$. Then, it reaches B through TT_k, the closest to B. The path can be found in Figure 10. □

Table 2 summarizes the number of nodes and degrees, diameter, and networost of fixed degree graphs and TG_n. Table 3 compares the network costs of TG_n and fixed degree graphs when the number of nodes is the same. Figure 11 verifies that TG_n has the lowest increase rate in network cost compared to other interconnection networks. This indicates that the network cost of TG_n improves more than that of other fixed degree graphs. In particular, when the network cost of other interconnection networks is more than 300, TG_n has 150, which improves the network cost by more than 50% compared to the existing mesh, honeycomb mesh, torus, and SEP.

IV. CONCLUSION

In this study, a new graph TG_n, which has a fixed degree of three based on a full binary tree, was proposed, and its properties, algorithms, and Hamiltonian cycles were analyzed. TG_n has an index that represents its depth, similar to a tree. However, it differs from a tree in that it has additional edges between the root node and the leaf nodes, as well as between the leaf nodes themselves. In particular, by connecting the root node and the leaf nodes with edges, the length of the path with a length of N is reduced to 1, thereby improving the diameter and network cost.

In this study, the diameter of TG_n was $2n - 2$ and the network cost was around $O(6n)$ through the routing algorithm. The most improved network cost can be achieved by comparing it with existing fixed degree graphs when the number of nodes is the same. The network cost of TG_n is about 50% better than the existing fixed degree graphs, showing excellent results. It was verified TG_n had the Hamiltonian cycle and three node-disjoint parallel paths. As a follow-up study on this study, a study that proves efficiency through symmetry, failure tolerance, and embedding will be needed.

REFERENCES

[1] N. G. Kim, "A study on the development of the composite measures of high-performance computer technology," Ph.D. thesis, Dept. Technol. Manag., Sungkyunkwan Univ., Seoul, South Korea, 2020.

[2] H. Lee, M. Kang, D. Kim, D. Seo, and Y. Li, "Epidemic vulnerability index for effective vaccine distribution against pandemic," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, early access, Aug. 15, 2022, doi: 10.1109/TCBB.2022.3198365.

[3] H. Lee and D. Seo, "FedLC: Optimizing federated learning in non-IID data via label-wise clustering," *IEEE Access*, vol. 11, pp. 42082–42095, 2023.

[4] H. Lee, I. Na, K. Bultakov, and Y. Kim, "A BPR-CNN based hand motion classifier using electric field sensors," *Comput., Mater. Continua*, vol. 71, no. 3, pp. 5413–5425, 2022.

[5] S. J. Ezell and R. D. Atkinson, "The vital importance of high-performance computing to U.S. competitiveness," ITIF, Washington, DC, USA, Apr. 2016.

[6] T. Huh, Y. Jung, and M. Koh, "Legal institutional improvement for activating national supercomputing ecosystem," *Korea Contents Soc.*, vol. 21, no. 2, pp. 641–651, 2021.

[7] J. S. Kim, "Connectivity and embedding algorithms among interconnection network HCN(n,n)," M.S. thesis, Graduate Comput. Sci., Sunchon Nat. Univ., Suncheon, South Korea, Tech. Rep., 2000.

[8] K. W. Doty, "New designs for dense processor interconnection networks," *IEEE Trans. Comput.*, vol. C-33, no. 5, pp. 447–450, May 1984.

[9] V. E. Mendia and D. Sarkar, "Optimal broadcasting on the star graph," *IEEE Trans. Parallel Distrib. Syst.*, vol. 3, no. 4, pp. 389–396, Jul. 1992.

[10] (Jun. 2023). *Highlights*. [Online]. Available: https://www.top500.org

[11] B. Parhami and M. Rakov, "Perfect difference networks and related interconnection structures for parallel and distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 8, pp. 714–724, Aug. 2005.

[12] J. W. Kwak, H. J. Ban, and C. S. Jhon, "Torus Ring: Improving performance of interconnection network by modifying hierarchical ring," *Parallel Comput.*, vol. 32, nos. 5–6, pp. 196–208, 2005.

[13] J. A. Bondy and U. S. R. Murty, *Graph Theory With Applications*, 5th ed. New York, NY, USA: Elsevier, 1982.

[14] S. B. Akers and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection networks," *IEEE Trans. Comput.*, vol. 38, no. 4, pp. 555–556, Apr. 1989.

[15] S. Lakshmivarahan, J.-S. Jwo, and S. K. Dhall, "Symmetry in interconnection networks based on Cayley graphs of permutation groups: A survey," *Parallel Comput.*, vol. 19, no. 4, pp. 361–407, Apr. 1993.

[16] J. H. Seo, "Petersen-torus interconnection network based on Petersen graph," Ph.D. thesis, Dept. Comput. Sci., Sunchon Univ., Suncheon, South Korea, 2008.

[17] V. S. Adve and M. K. Vernon, "Performance analysis of mesh interconnection networks with deterministic routing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 3, pp. 225–246, Mar. 1994.

[18] Y. Saad and M. H. Schultz, "Topological properties of hypercubes," *IEEE Trans. Comput.*, vol. C-37, no. 7, pp. 867–872, Jul. 1988.

[19] H. S. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.*, vol. C-20, no. 2, pp. 153–161, Feb. 1971.

[20] M. R. Samatham and D. K. Pradhan, "The de Bruijn multiprocessor network: A versatile parallel processing and sorting network for VLSI," *IEEE Trans. Comput.*, vol. 38, no. 4, pp. 567–581, Apr. 1989.

[21] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. San Mateo, CA, USA: Morgan Kaufmann, 1992.

[22] F. P. Preparata and J. E. Vuillemin, "The cube-connected cycles: A versatile network for parallel computation," *Commun. ACM*, vol. 24, no. 5, pp. 300–309, 1981.

[23] K. A. Efe, "A variation on the hypercube with lower diameter," *IEEE Trans. Comput.*, vol. 40, no. 11, pp. 1312–1316, Nov. 1991.

[24] F. Harary, *Graph Theory*. Reading, MA, USA: Addison-Wesley, 1969.

[25] S. L. Johnsson, "Communication efficient basic linear algebra computations on hypercube architectures," *J. Parallel Distrib. Comput.*, vol. 4, no. 2, pp. 133–172, Apr. 1987.

[26] S. B. Akers and B. Krishnamurthy, "On group graphs and their fault tolerance," *IEEE Trans. Comput.*, vol. C-36, no. 7, pp. 885–888, Jul. 1987.

[27] Z. T. Chou, C.-C. Hsu, and J.-P. Shue, "Bubblesort star graphs: A new interconnection network," in *Proc. 9th Int. Parallel Process. Symp.*, 1996, pp. 41–48.

[28] S. Latifi, M. M. D. Azevedo, and N. Bagherzadeh, "The star connected cycles: A fixed-degree network for parallel processing," in *Proc. Int. Conf. Parallel Process.*, Aug. 1993, pp. 91–95.

[29] S. Ranka, J. C. Wang, and N. Yeh, "Embedding meshes on the star graph," *J. Parallel Distrib. Comput.*, vol. 19, no. 2, pp. 131–135, Oct. 1993.

[30] C. Wei-Kuo and C. Rong-Jaye, "The (n, k)-star graph: A generalized star graph," *Inf. Process. Lett.*, vol. 56, no. 5, pp. 259–264, Dec. 1995.

[31] C.-H. Yeh and E. M. Varvarigos, "Macro-star networks: Efficient low-degree alternatives to star graphs for large-scale parallel architectures," in *Proc. 6th Symp. Frontiers Massively Parallel Comput.*, Oct. 1996, pp. 290–297.

[32] J.-S. Jwo, S. Lakshmivarahan, and S. K. Dhall, "A new class of interconnection networks based on the alternating group," *Networks*, vol. 23, no. 4, pp. 315–326, Jul. 1993.

[33] V. Bokka, H. Gurla, S. Olariu, and J. L. Schwing, "Podality-based time-optimal computations on enhanced meshes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 8, no. 10, pp. 1019–1035, Oct. 1997.

[34] A. G. Ranade and S. L. Johnsson, "The communication efficiency of meshes, Boolean cubes and cube connected cycles for wafer scale integration," in *Proc. Int. Conf. Parallel Process.*, 1987, pp. 479–482.

[35] M.-S. Chen, K. G. Shin, and D. D. Kandlur, "Addressing, routing, and broadcasting in hexagonal mesh multiprocessors," *IEEE Trans. Comput.*, vol. 39, no. 1, pp. 10–18, Jan. 1990.

[36] S. Latifi and P. K. Srimani, "A new fixed degree regular network for parallel processing," in *Proc. 8th IEEE Symp. Parallel Distrib. Process.* Washington, DC, USA: IEEE Computer Society, Oct. 1996, pp. 152–159.

[37] I. Stojmenovic, "Honeycomb networks: Topological properties and communication algorithms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 8, no. 10, pp. 1036–1042, Oct. 1997.

[38] K. W. Tang and S. A. Padubidri, "Diagonal and toroidal mesh networks," *IEEE Trans. Comput.*, vol. 43, no. 7, pp. 815–826, Jul. 1994.

[39] B. O. Seong, "NSEP interconnection network improving the network cost of SEP," M.S. thesis, Dept. Comput. Educ. Inform., Sunchon Univ., Suncheon, South Korea, 2021.

[40] J. W. Robin and J. W. John, *Graphs: An Introductory Approach*. USA: Canada and United States of America, 1990.

[41] K. Day and A. Tripathi, "A comparative study of topological properties of hypercubes and star graphs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 1, pp. 31–38, Jan. 1994.

[42] L. N. Bhuyan and D. P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Trans. Comput.*, vol. C-33, no. 4, pp. 323–333, Apr. 1984.

BO OK SEONG received the M.S. degree in computer education information from Sunchon National University, Chonnnam, Republic of Korea, in 2021. She is currently a Researcher with the Robot Science Education Center, Sunchon National University, South Korea. Her research interests include parallel computer, graph theory, interconnection networks, data science, and deep learning.

JIN-HYEOK JANG is currently a high school student and participated in the research as a student with the Institute of Science and Gifted Education, Suncheon National University, South Korea. His research interests include parallel computers, graph theory, and interconnected networks.

HYEONG OK LEE received the Ph.D. degree in computer science from Chonnam National University, Chonnam, Republic of Korea, in 1999. He is currently a Professor in computer education and the Director of the Robot Science Education Center, Sunchon National University, South Korea. His research interests include parallel computer, graph theory, interconnection networks, robot science, and robot education.