mathematics

MDPI

Article

Dual Protection Routing Trees on Graphs

Kung-Jui Pai

Department of Industrial Engineering and Management, Ming Chi University of Technology,
New Taipei City 24301, Taiwan; poter@mail.mcut.edu.tw

Abstract: In IP networks, packet forwarding is destination-based and hop-by-hop, and routes are built as needed. Kwong et al. introduced a protection routing in which packet delivery to the destination node can proceed uninterrupted in the event of any single node or link failure. He then showed that "whether there is a protection routing to the destination" is NP-complete. Tapolcai found that two completely independent spanning trees, abbreviated as CISTs, can be used to configure the protection routing. In this paper, we proposed dual protection routing trees, denoted as dual-PRTs to replace CISTs, which are less restrictive than CISTs. Next, we proposed a transformation algorithm that uses dual-PRTs to configure the protection routing. Taking complete graphs K_n, complete bipartite graphs $K_{m,n}$, hypercubes Q_n, and locally twisted cubes LTQ_n as examples, we provided a recursive method to construct dual-PRTs on them. This article showed that there are no two CISTs on $K_{3,3}$, Q_3, and LTQ_3, but there exist dual-PRTs that can be used to configure the protection routing. As shown in the performance evaluation of simulation results, for both Q_n and LTQ_n, we get the average path length of protection routing configured by dual-PRTs is shorter than that by two CISTs.

Keywords: protection routing; completely independent spanning trees; dual protection routing trees; complete graphs; complete bipartite graphs; hypercubes; locally twisted cubes

MSC: 05C90; 68R10

check for updates

Citation: Pai, K.-J. Dual Protection Routing Trees on Graphs. *Mathematics* **2023**, *11*, 3255. https://doi.org/10.3390/math11143255

Academic Editor: Andrea Scozzari

Received: 5 June 2023
Revised: 8 July 2023
Accepted: 19 July 2023
Published: 24 July 2023

1. Introduction

In IP networks, intra-domain routing has traditionally relied on distributed computing among routers. Packet forwarding is destination-based and hop-by-hop, and routes are created when needed by a centralized unit. Kwong et al. [1] introduced a reactive routing scheme called protection routing, which employs the multi-paths technique for packet forwarding. If the routing is a protection routing, packet delivery to the destination node can proceed uninterrupted in the presence of any single node or link failure. All routers have an alternate path they can use in case of failure to forward packets.

Let an undirected simple graph $G = (V(G), E(G))$, where the vertex set $V(G)$ and the edge set $E(G)$ represent the set of nodes and the set of communication links between nodes, respectively. Let $k \geq 2$ be an integer and $T_1, T_2, \ldots, T_k$ be spanning trees of a graph G. A vertex in a tree T_i is a leaf if its degree is one, and an inner vertex otherwise. Two paths from u to v are internally vertex-disjointed if they have no common vertices other than u and v. Next, two spanning trees T_i and T_j are edge-disjointed if they share no common edge. Then, the spanning trees $T_1, T_2, \ldots, T_k$ are completely independent spanning trees, abbreviated as CISTs, if they are pairwise edge-disjointed and internally vertex-disjointed. The k-CISTs problem was first posed in 2001 by Hasunuma [2]. Then, Tapolcai [3] provided sufficient conditions for protection routing in IP networks according to two CISTs. He gave a deterministic polynomial construction, which builds up a protection routing for an arbitrary destination node by two CISTs.

Previous related works are described below. Hasunuma first worked on the theoretical study of CISTs and showed that the underlying graph of any k-connected line-directed graph recognizes k CISTs [2], and there exist two CISTs in a four-connected maximal planar graph [4] and the Cartesian product of two two-connected graphs [5]. Cheng et al. constructed two CISTs in crossed cubes [6]. Darties et al. determined three CISTs in some Cartesian products of three cycles [7]. Pai et al. proposed several known CISTs, such as complete graphs, complete bipartite, complete tripartite graphs [8], and variants of hypercubes [9]. Tapolcai [3] showed that two CISTs have an application for configuring protection routing. According to this, Pai et al. construct protection routings via CISTs on dense Gaussian on-chip networks [10].

In addition to protection routing, there are studies on other fault-tolerant routing strategies. By using the Hopfield neural network, Samavi and Khadivi proposed a fault avoidance routing method that keeps the transmission message as far away from the faulty nodes as possible [11]. Bossard and Kaneko proposed a node-to-node routing algorithm in an n-dimensional k-ary torus that is tolerant to faults, and this method also tolerates faulty node clusters [12]. For the r-dimensional generalized hypercube, Shu et al. studied three-component connectivity and proposed a fault-tolerant routing algorithm based on it [13]. This method can construct at least one fault-free path between any two distinct fault-free vertices. Romanov et al. used circulant topologies as a promising deadlock-free topology for on-chip networks and proposed a deadlock-free routing algorithm based on it [14].

Recently, we found that the protection routing can still be constructed by reducing some of the edge-disjoint requirements of two CISTs. Therefore, the novelty of this paper is that we defined new trees called dual protection routing trees (abbreviated as dual-PRTs) and devised a transformation algorithm to configure protection routing through them. Taking complete graphs, complete bipartite graphs, hypercubes, and locally twisted cubes as examples, we provided recursive methods on which to construct dual-PRTs and configure protection routings. Furthermore, we supplemented some analysis on simulation to evaluate the corresponding performance. The performance evaluation showed that dual-PRTs are better than two CISTs.

The rest of the paper is organized as follows: Section 2 introduces the necessary definitions and theorems for protection routing and CISTs. In Section 3, we first propose definitions of dual-PRTs and a transformation algorithm that uses dual-PRTs to configure the protection routing. Taking complete graphs, complete bipartite graphs, hypercubes Qn, and locally twisted cubes as examples, we provided a recursive method to construct dual-PRTs on them. Next, in order to compare with previous related research, we made the performance assessment of protection routings on hypercubes and locally twisted cubes in Section 4. Finally, Section 5 is the conclusion of this paper.

2. Preliminaries

The topology of a network is usually modeled as an undirected graph G = (V(G), E(G)). The neighborhood of a vertex v in a graph G, denoted by $N_G(v)$, is the set of vertices adjacent to v in G. In a tree, an edge is called a leaf edge (respectively, a stem) if it is adjacent to at least one leaf (respectively, two inner vertices). For convenience, the terms “networks” and “graphs”, “nodes” and “vertices”, and “links” and “edges” are often used interchangeably in this paper. For a destination node $d \in V(G)$, the route of traffic destined for d is a directed acyclic graph R_d = (V(G), $E_d(G)$), where the underlying graph of R_d is the spanning subgraph of G and each node $u \in V(G)\backslash\{d\}$ has at least one outgoing link in R_d. We use <u, v> to denote a directed link from u to v. If <u, v> $\in E_d(G)$, node v is called the primary next-hop of node u, or PNH for short, and the link <u, v> is called a primary link of u. Furthermore, node u is called upstream of node v if there exists a directed path from u to v in R_d. In contrast, node v is downstream of node u. Except for the destination node d, when a single component fails $f \in V(G) \cup E_d(G)\backslash\{d\}$, let $G - f$ and $R_d - f$ be the residual network and routing obtained from G and R_d, respectively. Kwong et al. [1] gave the following two definitions.

Definition 1 ([1]). *A node $u \in V(G)\backslash\{d\}$ in a routing R_d is said to be protected with respect to d if after any single component failure $f \in V(G) \cup E_d(G)\backslash\{d\}$ that affects node u's PNH, then there exists a node $x \in N_G(u) - f(u)$ such that the following two conditions (1) x is not upstream of u in $R_d - f$; (2) x and all its downstream nodes, except d, have at least one PNH in $Rd - f$.*

Definition 2 ([1]). *A routing R_d is a protection routing if every node $u \in V(G)\backslash\{d\}$ is protected in R_d. A network G is protectable if there exists a protection routing R_d for all $d \in V(G)$, otherwise, G is unprotectable.*

In fact, x is called the second next-hop of node u, SNH for short, and is denoted by SNH(u). We notice that when a failure occurs, the candidate SNHs of node u are not unique. However, the first condition is to avoid forwarding loops in case of failure, and the second condition guarantees that packets are delivered to d through node x and its downstream nodes in $R_d - f$. For example, Figure 1 shows a protection routing in hypercube Q_3 (we formally introduce it in Section 3.3) for the destination node 0, and Table 1 gives a candidate SNH of node $u \in V(Q_3)\backslash\{0\}$.

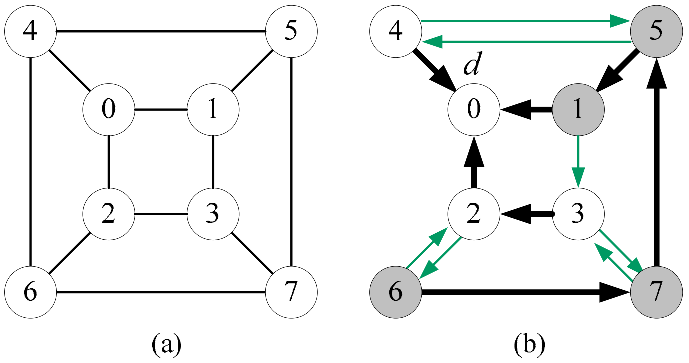


Figure 1. (**a**) Hypercube Q_3 and (**b**) a protection routing for destination node 0 in Q_3, where thick arcs indicate primary links and thin arcs indicate alternative links.

Table 1. Candidate SNHs of node $u \in V(Q_3)\backslash\{0\}$.

Component Failure	Second Next-Hop
<1, 0>	SNH(1) = 3
<2, 0>	SNH(2) = 6
<4, 0>	SNH(4) = 5
<3, 2> or 2	SNH(3) = 7
<5, 1> or 1	SNH(5) = 4
<6, 7> or 7	SNH(6) = 2
<7, 5> or 5	SNH(7) = 3

Then, Tapolcai provided sufficient conditions for protection routing in IP networks according to two CISTs. The following definition is about CISTs.

Definition 3 ([2]). *The spanning trees T_1 and T_2 are two completely independent spanning trees if they are inner-vertex-disjoint and edge-disjoint.*

Then, inner-vertex-disjoint and edge-disjoint are defined as follows:

Definition 4. *Two spanning trees T_1 and T_2 are inner-vertex-disjoint if any vertex can only be an inner node in T_1 or T_2.*

Definition 5. *Two spanning trees T_1 and T_2 are edge-disjoint if they share no common edge.*

3. Main Results

In [3], Tapolcai uses two CISTs to configure a protection routing for a destination node d. Then, we find another way to construct routing through two spanning trees T_1 and T_2 in which most of the leaf edges can be shared. We first give the following two definitions:

Definition 6. *Two spanning trees T_1 and T_2 are stem-disjoint if they share no common stem.*

Definition 7. *The spanning trees T_1 and T_2 are dual protection routing trees based on destination node d, which are denoted as dual-PRT^{d}s, if Definitions 4, 6 and the following condition are met:*

- *Without loss of generality, we assume that node d is an inner vertex in T_1. There exists an inner node d' in T_2 such that (d, d') is a leaf edge in T_2 and d' is adjacent to another inner node w ($\neq d$) in T_1.*

In this paper, we proposed a new tree structure in Definition 7. Similar to CISTs, spanning trees T_1 and T_2 are used to construct a protection routing. However, the difference from CISTs is that we only need stem-disjoint. Obviously, d is the root of the first tree. We need another node d' to be the root of the second tree. In order to connect the stems of the two trees, (d, d') must exist. Finally, d' must have a neighbor w ($\neq d$) as its second next-hop. For example, Figure 2 shows dual-PRT^{0}s. The inner nodes of T_1 are vertices 0, 2, 3, and 4, and the inner nodes of T_2 are vertices 1, 5, 6, and 7. T_1 and T_2 follow Definition 4. Then, the stem sets of T_1 and T_2 are {(0, 2), (0, 4), (2, 3)} and {(1, 5), (5, 7), (6, 7)}, respectively, and they follow Definition 6. Finally, let d', w be vertices 1 and 3, respectively. Since (0, 1) is a leaf edge in T_2 and vertex 1 is adjacent to inner node 3 in T_1, according to Definition 7, T_1 and T_2 are dual-PRT^{0}s. In fact, Figure 1b shows the protection routing configured by T_1 and T_2. We propose a formal construction algorithm later.

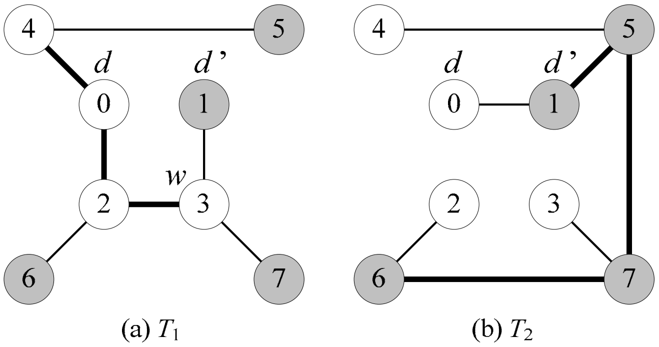


Figure 2. (**a**) T_1 and (**b**) T_2 are dual-PRT^{0}s where thick lines indicate stems and thin lines indicate leaf edges.

Lemma 1. *There are no two CISTs in hypercube Q_3.*

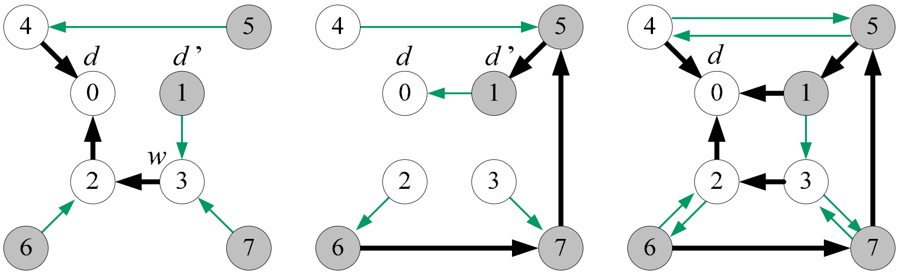
Proof. As shown in Figure 1a, hypercube Q_3 has eight nodes and twelve edges. A tree with eight nodes has seven edges. By Definition 5, we need fourteen edges to find two edge-disjoint spanning trees, but Q_3 has only twelve edges. □

According to Lemma 1, if there are no two CISTs in Q_3, the protection routing cannot be configured by Tapolcai's method [3]. That is why we propose dual-PRT^{d}s. Now we present the construction method in which the protecting routing is configured by dual-PRT^{d}s, as follows.

For example, taking Figure 2a,b as input, according to steps 1 and 2 of Algorithm 1, we have $T_{1'}$ and $T_{2'}$, as shown in Figure 3a,b. Next, classify the stems and leaf edges of the two trees into primary links or alternative links according to steps 3 and 4, and then a protection routing R_0 can be obtained, as shown in Figure 3c.

Algorithm 1: Configuring a protection routing via dual-PRT^{d}s

```
Input: Dual-PRT$^d$s $T_1$, $T_2$ of a network $G$ where $d$ is the destination node.
Output: A protection routing $R_d = (V(G), PL, AL)$ where $PL$ is the primary links set and $AL$ is the
        alternative links set
Step 1  $T_{1'} \leftarrow T_1$ which takes all links directed to root $d$;
Step 2  $T_{2'} \leftarrow T_2$ which takes all links directed to root $d$;
Step 3  $PL \leftarrow$ all stems in $T_{1'} \cup$ all stems in $T_{2'} \cup \{<d', d>\}$;
Step 4  $AL \leftarrow$ all leaf edges in $T_{1'} \cup$ all leaf edges in $T_{2'}$ except $<d', d>$;
Step 5  If a vertex is a leaf in both $T_1$ and $T_2$, then add its $T_1$ leaf edge to $PL$ and add its $T_2$ leaf
        edge to $AL$;
Step 6  Return $R_d = (V(G), PL, AL)$
```



(a) T_1'

(b) T_2'

(c)

Figure 3. (**a**) $T_{1'}$ and (**b**) $T_{2'}$ are all links directed to root 0 from T_1 and T_2, respectively, (**c**) a protection routing R_0, where thick arcs indicate primary links and thin arcs indicate alternative links.

According to Definition 1, a routing R_d is a protection routing if every node except d is protected in R_d. In order to prove that the routing constructed by Algorithm 1 is a protection routing, we give the following theorem:

Theorem 1. *The routing constructed by Algorithm 1 is a protection routing with respect to node d.*

Proof. According to steps 1 and 2 in Algorithm 1, all edges of T_1 and T_2 are directed to destination node d. Obviously, stems of T_1 and stems of T_2 respectively form 2 primary routes by step 3. Then $<d', d>$ connect the second route to destination d. Figure 3 can be used as an illustration. By Definition 6, stem-disjoint can ensure that two primary routes will not overlap. Additionally, according to Definition 4, each vertex can only be an inner node in T_1 or T_2, which ensures that each vertex has a primary link and an alternative link. If a vertex is a leaf in both T_1 and T_2, it still has both links at step 5. Since $<d', d>$ is a primary link, d' needs another link $<d', w>$ as an alternative link. Now, let us consider a component failure, it could be a node or a link. If it is a node (respectively, a link), we assume it is the node z (respectively, $<u, z>$) and $z = \text{PNH}(u)$. Regardless of whether the failure is z or <u, z>, the link $<u, z>$ is unavailable. If $<u, z>$ is in T_1 (respectively, T_2), then node u's alternative link is in T_2 (respectively, T_1), so it forwards the packet to the primary routes of T_2 (respectively, T_1) to the destination node d. Therefore, the routing constructed by Algorithm 1 is a protection routing with respect to node d. □

In [8], we construct CISTs on complete graph K_n where $n \geq 4$, complete bipartite graph $K_{n,m}$ where $m \geq n \geq 4$. Therefore, we construct dual-PRT^{d}s on them, and make some comparisons.

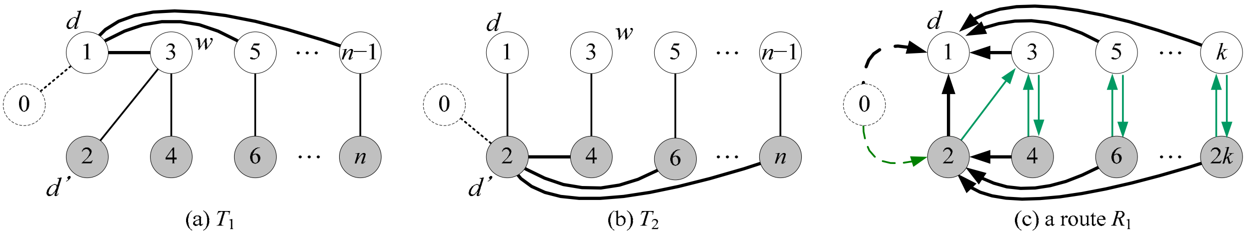
3.1. Dual-PRT^{d}s on Complete Graphs

A complete graph with n vertices denoted as K_n is a graph in which every pair of distinct vertices is connected by a unique edge.

Theorem 2. *There exist dual-PRT^{d}s in K_n where d is any vertex of K_n and $n \geq 4$.*

Proof. If n is even (respectively, odd), let the vertices of K_n be labeled 1, 2, 3, . . ., n (respectively, 0, 1, 2, 3, . . ., n). For the case that n is even, let all odd vertices be inner nodes in T_1 and all even vertices be inner nodes in T_2. Stems are (1, 3), (1, 5), . . ., (1, $n-1$) (respectively, (2, 4), (2, 6), . . ., (2, n)) and leaf edges are (3, 2), (3, 4), (5, 6), . . ., ($n-1$, n) (respectively, (1, 2), (3, 4), . . ., ($n-1$, n)) in T_1 (respectively, T_2). Clearly, T_1 and T_2 obey Definitions 4 and 6. Let d = 1, d' = 2, w = 3, and then Definition 7 holds. T_1 and T_2 are dual-PRT^{1}s in K_n. In the case that n is odd, the trees are similar to the ones in the case that n is even. We let additional vertex 0 be adjacent to vertex 1 and 2 in T_1 and T_2, respectively. Clearly, T_1 and T_2 still obey Definition 7, and they are dual-PRT^{1}s in K_n. It is widely known that K_n is vertex-symmetric, so there exist dual-PRT^{d}s where d can be any vertex in K_n. □

Figure 4a,b illustrate the proof of Theorem 2. The tree diameter is the length of the shortest path between the most distanced nodes, which affects the length of the routing in practice. In [8], the diameters of CISTs in K_n are all 3, and the diameters of our dual-PRT^{d}s in K_n are 4, which is slightly worse than the former.



(a) T_1 (b) T_2 (c) a route R_1

Figure 4. (**a**) T_1 and (**b**) T_2 are dual-PRT^{1}s, (**c**) protection routing R_1, where thick arcs indicate primary links and thin arcs indicate alternative links.

Theorem 3. *K_n is protectable for $n \geq 4$.*

Proof. According to Theorem 2, there exist dual-PRT^{d}s in K_n for $n \geq 4$. Then, by Algorithm 1, we can use dual-PRT^{d}s to configure the protection routing as shown in Figure 4c. According to Definition 2, this theorem holds. □

3.2. Dual-PRT^{d}s on Complete Bipartite Graphs

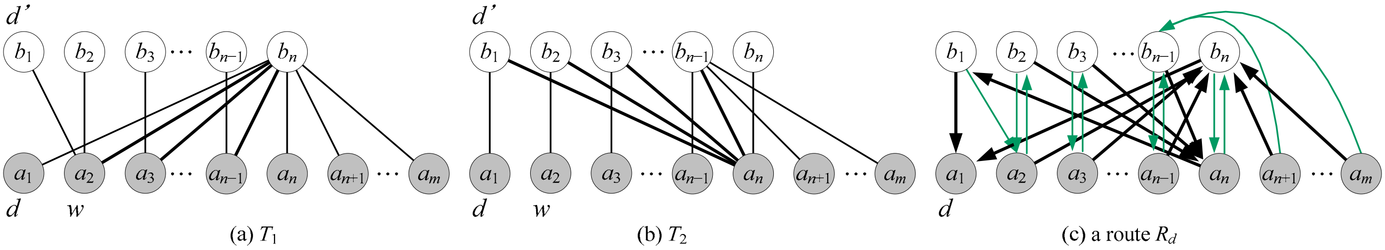
A complete bipartite graph, denoted by $K_{m,n}$ is a graph whose vertices can be partitioned into two subsets V_1 and V_2 such that $|V_1| = m$, $|V_2| = n$ and two vertices u and v are adjacent if and only if $u \in V_1$ and $v \in V_2$.

Theorem 4. *There exist dual-PRT^{d}s in $K_{m,n}$ where d is any vertex in* V_1 *(or* V_2*) and $m \geq n \geq 3$.*

Proof. Without loss of generality, we assume that $|V_1| = m \geq |V_2| = n$. Let the vertices of V_1 (respectively, V_2) be labeled $a_1, a_2, a_3, \ldots, a_{\mathrm{m}}$ (respectively, $b_1, b_2, b_3, \ldots, b_{\mathrm{n}}$). In the case that $d \in V_1$, let vertices $a_2, a_3, \ldots, a_{n-1}, b_n$ be inner nodes in T_1 and vertices $b_1, b_2, \ldots, b_{n-1}$ be inner nodes in T_2. Stem edges are (a_2, b_n), (a_3, b_n), . . ., (a_{n-1}, b_n) (respectively, (b_1, a_n), (b_2, a_n), . . ., (b_{n-1}, a_n)) and leaf edges are (b_1, a_2), (b_2, a_2), (b_3, a_3), . . ., (a_n, b_n), (a_1, b_n), (a_{n+1}, b_n), (a_{n+2}, b_n), . . ., (a_m, b_n) (respectively, (a_1, b_1), (a_2, b_2), . . ., (b_n, a_n), (a_{n+1}, b_{n-1}), (a_{n+2}, b_{n-1}), . . ., (a_m, b_{n-1})) in T_1 (respectively, T_2). Clearly, T_1 and T_2 obey Definitions 4 and 6. Let $d = a_1$, $d' = b_1$, $w = a_2$, then Definitions 7 holds. Since a_1 can be replaced by any vertex in V_1, T_1 and T_2 are dual-PRT^{d}s in $K_{m,n}$ where $d \in V_1$. For the case that $d \in V_2$, we can swap V_1 and V_2, and prove it in a similar way. □

Figure 5 is used to illustrate the proof of Theorem 4. In [8], there are no two CISTs in $K_{3,3}$, because two edge-disjoint spanning trees require a total of 10 edges while $K_{3,3}$ only has 9 edges. However, $K_{3,3}$ has dual-PRT^{d}s, which can be used to configure a protection routing.

Next, when $m \geq n \geq 4$ the diameter of CISTs in $K_{m,n}$ are all 5 in [8], and the diameters of our dual-PRT^{d}s in $K_{m,n}$ are 4, which is slightly better than the former.



(a) T_1
(b) T_2
(c) a route R_d

Figure 5. (**a**) T_1 and (**b**) T_2 are dual-PRT^{d}s, (**c**) a protection routing R_d, where thick arcs indicate primary links and thin arcs indicate alternative links.

According to Theorem 4, there exist dual-PRT^{d}s in $K_{m,n}$ where $d \in V_1$ (or V_2). They can be used to configure the protection routing via Algorithm 1. Finally, by Definition 2, we have the following theorem:

Theorem 5. *$K_{m,n}$ is protectable where $m \geq n \geq 3$.*

Theorem 5 improves the known result that $K_{m,n}$ is protectable where $m \geq n \geq 4$.

3.3. Dual-PRT^{d}s on Hypercubes

The n-dimensional hypercube, denoted by Q_n, is a graph with 2^n vertices such that each vertex corresponds to an n-tuple $(b_n, b_{n-1}, \ldots, b_1)$ on the set $\{0, 1\}^n$ and two vertices are adjacent by an edge if and only if they differ in exactly one coordinate [15]. For example, Q_3 is shown in Figure 1a. For conciseness, the labels of vertices are changed to their decimal. The hypercube is one of the most popular interconnection networks because of its attractive properties, including regularity, vertex symmetricality, edge symmetric, small diameter, strong connectivity, recursive construction, partition capability, and small link complexity.

Lemma 2. *Q_3 has dual-PRT^{d}s with a diameter is 5 where d is any vertex in Q_3.*

Proof. Let destination node d be vertex 0. Dual-PRT^{0}s are shown in Figure 2. Clearly, the diameters of two trees are both 5. Vertices 0, 2, 3, and 4 are inner nodes in T_1, and vertices 1, 5, 6, and 7 are inner nodes in T_2. Stems are (0, 2), (0, 4), (2, 3) (respectively, (1, 5), (5, 7), (6, 7)), and leaf edges are (1, 3), (3, 7), (2, 6), (4, 5) (respectively, (0, 1), (2, 6), (3, 7), (4, 5)) in T_1 (respectively, T_2). Clearly, T_1 and T_2 obey Definitions 4 and 6. Let d' be vertex 1 and w be vertex 3, and then Definition 7 is met. Since Q_n is vertex-symmetric and there exists an automorphism of Q_n, vertex 0 can be replaced by any vertex in Q_3. This lemma holds. □

In [9], we provide and prove a simple unified approach to construct two CISTs in the n-dimensional hypercubes variants (abbreviated as QV_n) using two CISTs in QV_{n-1}. This approach is also applicable for dual-PRT^{d}s. For QV_n, it is constructed recursively from two QV_{n-1}s. Then $(n-1)$-dimensional dual-PRT^{d}s also exist in each QV_{n-1}. We can select a pair of port vertices and add an edge to connect two trees, where port vertices obey (1) removing the leftmost bit, and their labels will be the same; (2) there exists an edge between them in QV_n. For example, Figure 6 shows dual-PRT^{0}s of Q_4 which are constructed from dual-PRT^{0}s of Q_3. Clearly, port vertices of T_1 (respectively, T_2) are vertices 0 and 8 (respectively, 5 and 13).

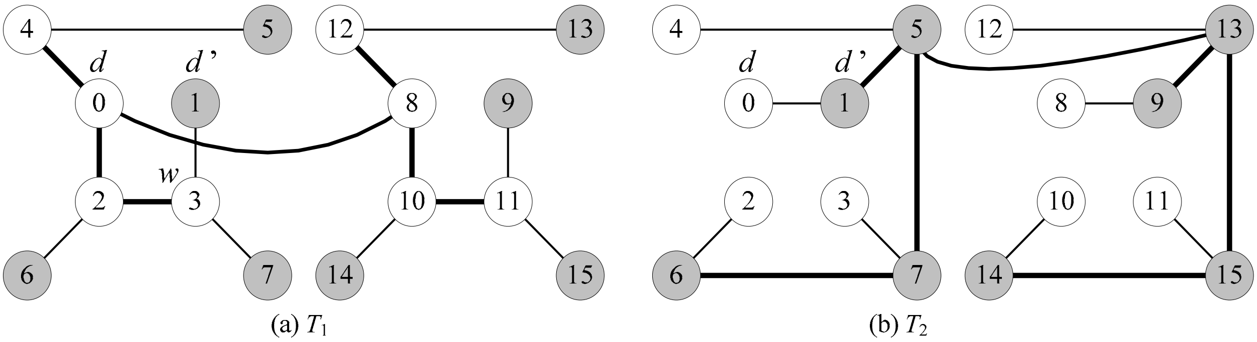


Figure 6. (**a**) T_1 and (**b**) T_2 are dual-PRT^{0}s of Q_4.

Using the unified approach, if we choose vertices near the center of the trees as their port vertices, then we can obtain a better result of the diameter of dual-PRT^{d}s for the construction. Suppose that T_1 and T_2 are dual-PRT^{d}s of QV_{n-1} and let T'_1 and T'_2 be dual-PRT^{d}s of QV_n. If a vertex u is a center vertex of T_1 and v is a center vertex of T_2, then the choice of a pair $(u, u + 2^{n-1})$ in T'_1 and a pair $(v, v + 2^{n-1})$ in T'_2 as port vertices can build two dual-PRT^{d}s of QV_n by induction on n. Since we can always choose the center vertices of T_1 and T_2 as port vertices, it follows that $D(T'_j) = 2 \cdot \lceil D(T_j)/2 \rceil + 1$ where $j \in \{1, 2\}$ and $D(T)$ denotes the diameter of the tree T. For example, as shown in Figures 2 and 6, the diameters of dual-PRT^{d}s of Q_3 are both 5 and the diameters of dual-PRT^{d}s of Q_4 are both 7. Then, by the unified approach and Lemma 2, we have the following theorem:

Theorem 6. *Q_n has dual-PRT^{d}s with a diameter is $2n - 1$ where $d \in V(Q_n)$ and $n \geq 3$.*

By Theorem 6, there exist dual-PRT^{d}s in Q_n where $d \in V(Q_n)$. According to Algorithm 1, two trees can be used to configure the protection routing. Then, by Definition 2, we have the following theorem:

Theorem 7. *Q_n is protectable where $n \geq 3$.*

As stated in Lemma 1, Theorem 7 improves on the known result that Q_n is protectable when $n \geq 4$. As far as we know, the diameters of two CISTs of Q_4 are both 8 [5]. The diameters of dual-PRT^{d}s of Q_n are one less than the diameters of two CISTs of Q_n while $n \geq 4$.

3.4. Dual-PRT^{d}s on LTQs

The n-dimensional locally twisted cube, denoted by LTQ_n, is defined recursively as follows (see [16]):

(1) LTQ_1 is the complete graph on two vertices labeled 0 and 1. LTQ_2 is a graph consisting of four vertices with labels 00, 01, 10, and 11 together with four edges (00, 01), (00, 10), (01, 11), and (10, 11);

(2) For $n \geq 3$, LTQ_n is composed of two subcubes $LTQ^0{}_{n-1}$ and $LTQ^1{}_{n-1}$ such that each vertex $x = 0b_{n-1}b_{n-2}\cdots b_1 \in V(LTQ^0{}_{n-1})$ is connected with the vertex $1(b_{n-1}\oplus b_1)b_{n-2}\cdots b_1 \in V(LTQ^1{}_{n-1})$ by an edge where $\oplus$ represents exclusivity.

LTQ_n is a variant of Q_n, and one advantage of LTQ_n is that the diameter is only about half of the diameter of Q_n. For example, LTQ_3 is shown in Figure 7a, its diameter is 2, while the value of Q_3 is 3. In [17], LTQ_n is vertex-transitive if and only if $n \leq 3$.

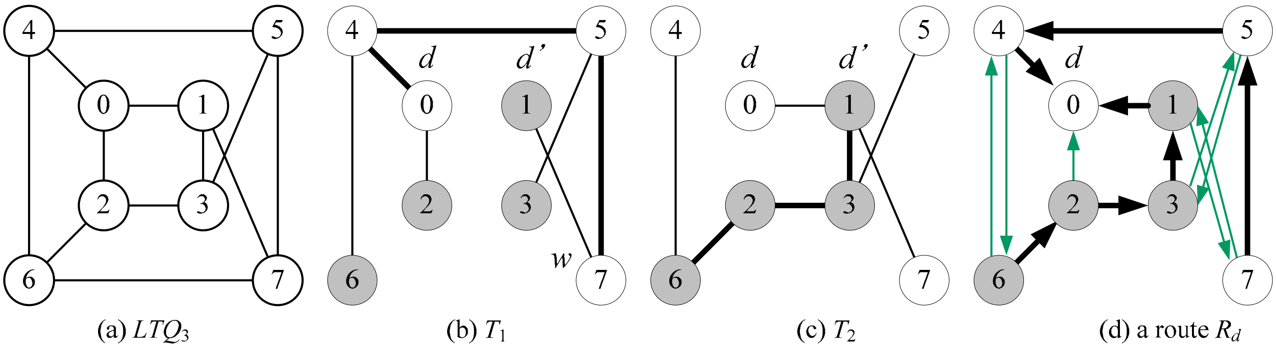


Figure 7. (**a**) LTQ_3 (**b**) T_1 and (**c**) T_2 are dual-PRT^{0}s of LTQ_3, (**d**) a protection routing R_d, where thick arcs indicate primary links and thin arcs indicate alternative links.

Lemma 3. *LTQ_3 has dual-PRT^{d}s with a diameter of 5 where d is any vertex in LTQ_3.*

Proof. Let destination node d be vertex 0. Dual-PRT^{0}s are shown in Figure 7b,c. Clearly, the tree diameters are both 5. Vertices 0, 4, 5, and 7 are inner nodes in T_1, and vertices 1, 2, 3, and 6 are inner nodes in T_2. Stems are (0, 4), (4, 5), (5, 7) (respectively, (1, 3), (2, 3), (2, 6)), and leaf edges are (0, 2), (1, 7), (3, 5), (4, 6) (respectively, (0, 1), (1, 7), (3, 5), (4, 6)) in T_1 (respectively, T_2). Clearly, T_1 and T_2 obey Definitions 4 and 6. Let d' be vertex 1 and w be vertex 7, and then Definition 7 holds. Since LTQ_3 is vertex-symmetric and there exists an automorphism of LTQ_3, vertex 0 can be replaced by any vertex in LTQ_3. This lemma holds. □

In [17], the full automorphism group of LTQ_n with $n \geq 4$ has exactly two orbits, and the odd and even vertices belong to the same group. We present dual-PRT^{d}s for d belonging to odd and even vertices, respectively.

Lemma 4. *LTQ_4 has dual-PRT^{d}s with diameters of 6 and 8 where d is any even vertex in LTQ_4.*

Proof. Let destination node d be vertex 0. Dual-PRT^{0}s are shown in Figure 8. Clearly, the diameters of the two trees are 8 and 6, respectively. Vertices 0, 3, 4, 5, 8, 9, 12, and 15 are inner nodes in T_1, and vertices 1, 2, 6, 7, 10, 11, 13, and 14 are inner nodes in T_2. Stems are (0, 4), (0, 8), (3, 5), (4, 5), (4, 12), (8, 9), (9, 15) (respectively, (1, 7), (1, 13), (2, 6), (6, 7), (6, 14), (7, 11), (10, 11)) and leaf edges are (0, 2), (1, 3), (4, 6), (5, 7), (8, 10), (9, 11), (12, 14), (13, 15) (respectively, (0, 1), (2, 3), (4, 6), (5, 7), (8, 10), (9, 11), (12, 14), (13, 15)) in T_1 (respectively, T_2). Clearly, T_1 and T_2 obey Definitions 4 and 6. Let d' be vertex 1 and w be vertex 3, and then Definition 7 holds. Since all even vertices belong to the same automorphism group in LTQ_4, this lemma holds. □

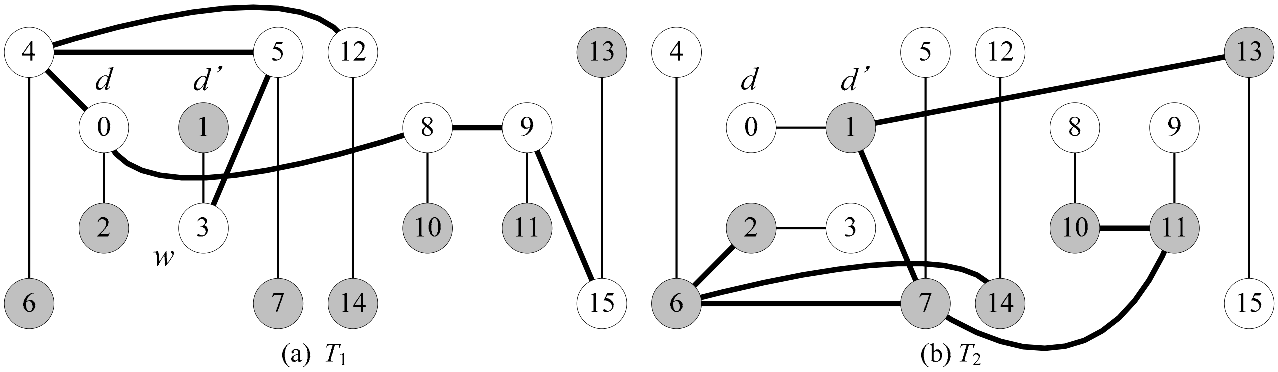


Figure 8. (**a**) T_1 and (**b**) T_2 are dual-PRT^{0}s of LTQ_4.

Lemma 5. *LTQ_4 has dual-PRT^{d}s with diameters of 6 and 8 where d is any odd vertex in LTQ_4.*

Proof. Let destination node d be vertex 1. Dual-PRT^{1}s are shown in Figure 9. Clearly, the diameters of the two trees are 6 and 8 respectively. Vertices 1, 2, 6, 7, 10, 11, 13, and 14 are inner nodes in T_1, and vertices 0, 3, 4, 5, 8, 9, 12, 15 are inner nodes in T_2. Stems are (1, 7), (1, 13), (2, 6), (6, 7), (6, 14), (7, 11), (10, 11) (respectively, (0, 4), (0, 8), (3, 5), (4, 5), (4, 12), (8, 9), (9, 15)) and leaf edges are (0, 2), (1, 3), (4, 6), (5, 7), (8, 10), (9, 11), (12, 14), (13, 15) (respectively, (0, 1), (2, 3), (4, 6), (5, 7), (8, 10), (9, 11), (12, 14), (13, 15)) in T_1 (respectively, T_2). Clearly, T_1 and T_2 obey Definitions 4 and 6. Let d' be vertex 0 and w be vertex 2, and then Definitions 7 holds. Since all odd vertices belong to the same automorphism group in LTQ_4, this lemma holds. □

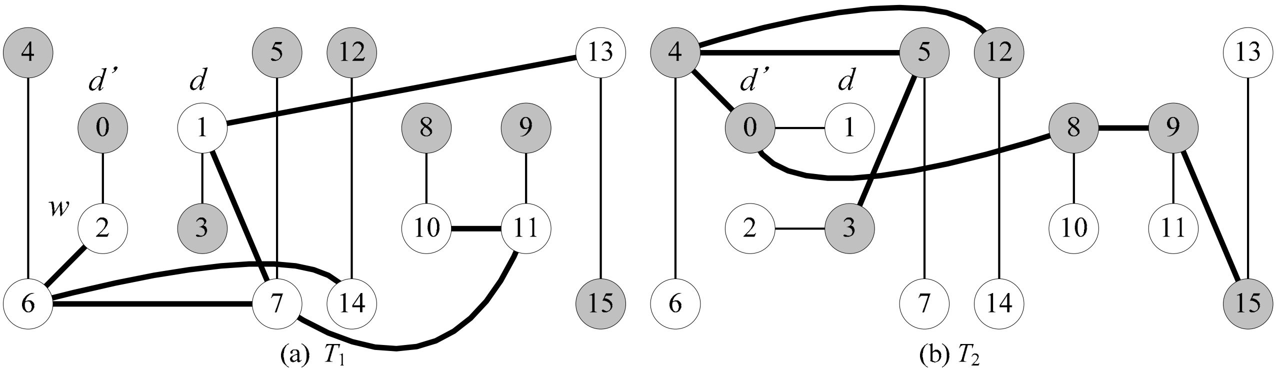


Figure 9. (**a**) T_1 and (**b**) T_2 are dual-PRT^{1}s of LTQ_4.

By using the unified approach [9] to configure dual-PRT^{d}s of LTQ_n from dual-PRT^{d}s of LTQ_{n-1}, we find that it is better to choose vertices near the center of the tree as port vertices. According to the edge adjacent rules of LTQ_n, it is necessary to select an even vertex x to ensure that $(x, x + 2^{n-1})$ exists. As a counterexample, $(1, 1 + 2^4)$ does not exist in LTQ_5. Then, we deal with the following theorem:

Theorem 8. *LTQ_n has dual-PRT^{d}s with the diameter $2n - 1$ (respectively, 6 or 8) while $n \geq 3$ and $n \neq 4$ (respectively, $n = 4$) where $d \in LTQ_n$.*

Proof. First, according to Lemmas 3, 4, and 5, this theorem holds when $n = 3$ or 4. Then, we use the unified approach [9] to recursively construct dual-PRT^{d}s of LTQ_{n+1} from dual-PRT^{d}s of LTQ_n while $n \geq 5$. We consider the following two cases:

Case 1. Destination node d is any even vertex in LTQ_n while $n \geq 5$.

We prove by induction, based on $n = 4$. Since LTQ_{n+1} consists of two LTQ_n, there are n-dimensional dual-PRT^{d}s in each LTQ_n. We adopt the dual-PRT$^{\mathrm{d}}$s shown in Figure 8 as the induction base. By using the unified approach, we choose vertex 0 of T_1 and vertex 6 of T_2 as port vertices. For LTQ_{n+1}, we connect a pair of port vertices $(0, 0 + 2^n)$ (respectively, $(6, 6 + 2^n)$)to link two T_1s (respectively, T_2s) in each subcube LTQ_{n-1}, and then we have dual-PRT$^{\mathrm{d}}$s T'_1 (respectively, T'_2) in LTQ_n. Because the length from vertex 0 (respectively, vertex 6) to the farthest leaf in T_1 (respectively, T_2) is 4, the diameter of recursively constructed dual-PRT$^{\mathrm{d}}$s is both $2n - 1$ while $n \geq 5$;

Case 2. Destination node d is any even odd in LTQ_n while $n \geq 5$.

The proof is similar to that of Case 1. In this case, we adopt the dual-PRT^{d}s shown in Figure 9 as the induction base and select vertex 6 of T_1 and vertex 0 of T_2 as port vertices. Other parts are the same as in case 1. □

For example, Figure 10 shows dual-PRT^{0}s on LTQ_5, which can be used to illustrate the proof of Theorem 8. According to Theorem 8, there exist dual-PRT^{d}s in LTQ_n where

$d \in V(LTQ_n)$ while $n \geq 5$. Then, two trees can be used to configure the protection routing via Algorithm 1. By Definition 2, we have the following theorem:

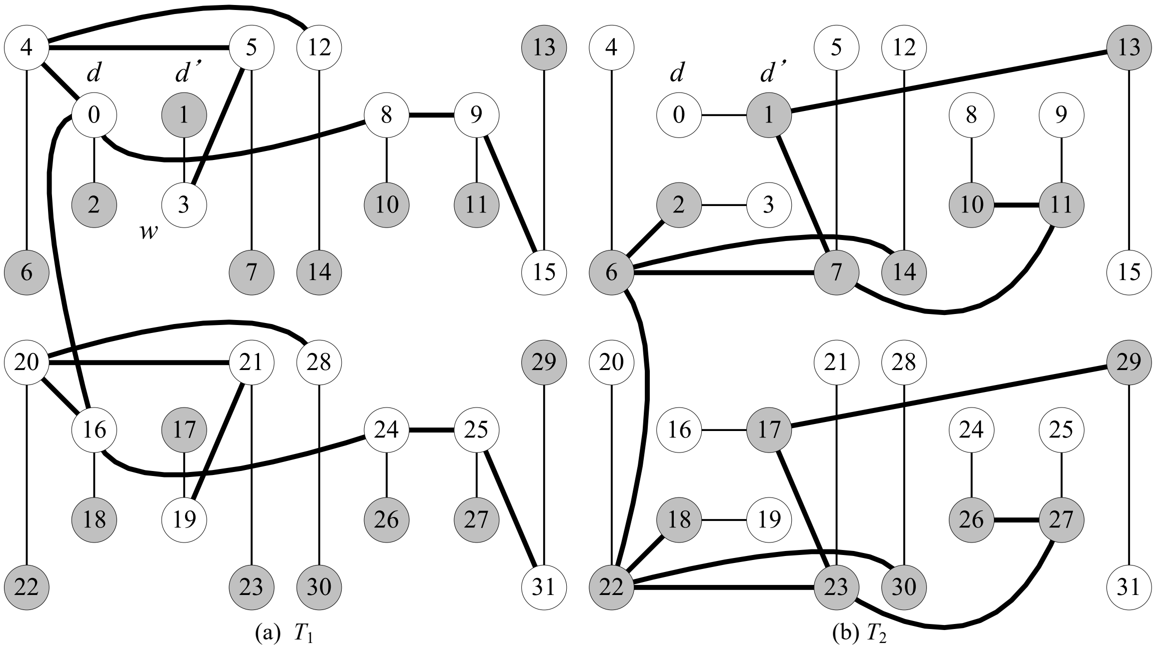


Figure 10. (**a**) T_1 and (**b**) T_2 are dual-PRT^{0}s of LTQ_5.

Theorem 9. *LTQ_n is protectable where $n \geq 3$.*

Theorem 9 improves the known result that LTQ_n is protectable when $n \geq 4$. As far as we know, the diameters of two CISTs of LTQ_4 are both 7 [9]. In each dimension of LTQ_n, the diameter of dual-PRT^{d}s is about the same as that of two CISTs.

4. Performance Evaluation

In this section, we present simulation results for evaluating the performance of the protection routings on Q_n and LTQ_n. The protection routings can be configured by dual-PRT^{d}s or two CISTs. For Q_n (respectively, LTQ_n), we used C programs to implement the routing algorithms by dual-PRT^{d}s in the previous section and two CISTs in [5] (respectively, [9]). To speed up the evaluation, we carried out the simulation separately for each of Q_n and LTQ_n by using a 5.10 GHz Intel® Core™ i9-12900 CPU and 32GB RAM under the Linux operating system.

For each dimension 3, 4, 5, …, 9, we randomly generated 1,000,000 instances of vertex-list (s, d, f) with $s \neq d \neq f$ for the two kinds of networks, where s, d, and f are the source, destination, and failure node of traffic, respectively. The protection routing R_d is constructed by dual-PRT^{d}s (respectively, two CISTs) using Algorithm 1 (respectively, Tapolcai's method [3]). We are interested in computing the path length from s to d under two scenarios: (i) no failure and (ii) a single node failure. For each instance, if no failure occurs, we compute the path length by tracking the primary links and PHNs alternately from s to d in R_d. For convenience, we call this specific routing the default path $P_{s,d}$. Then, the length of $P_{s,d}$ is defined by the number of links from s to d in R_d. Likewise, we are also concerned with computing path lengths in case of node failures in R_d. We assume that the appearance of node failure is uniformly distributed throughout the network. If failure node f = PNH(u) and f is contained in R_d, we calculate the path length of an alternate path $P_{s,u} \cup P_{u,\text{SNH}(u)} \cup P_{\text{SNH}(u),d}$. We then compute three statistical quantities related to path length: (a) average path length, (b) standard deviation of path length, and (c) maximum path length and the number of its occurrences. More descriptions and examples of the simulation process are available on a website [18] as Supplementary Materials.

Tables 2 and 3 show some simulation results of the two scenarios (no failure and a single node failure) on Q_n and LTQ_n. To save space, four additional tables are available on a website [18] as Supplementary Materials. In these tables, the three quantities mentioned above are calculated in the usual way according to statistics. They present simulation results for Q_n and LTQ_n when the routing is with two scenarios. Additionally, the ratio of invoking SNH under a single node failure is calculated (see the footer * in Tables 2 and 3). It is obvious that the ratio tends to decrease gradually as the network's dimension increases, and eventually, its effect on length becomes less pronounced. According to Lemma 1, there are no two CISTs in Q_3, so there are two missing values in Table 2. For the same reason, there are also two missing values in Table 3.

Table 2. Simulation results: the average path length on Q_n when the routing is with two scenarios: (1) no failure; and (2) a single node failure.

	No Failure		**A Single Node Failure**	
n	**2 CISTs**	**Dual-PRT^{d}s**	**2 CISTs**	**Dual-PRT^{d}s**
3		2.000		2.000 (* 16.69%)
4	3.233	2.533	3.394 (* 15.94%)	2.580 (* 10.98%)
5	4.272	3.031	4.445 (* 10.92%)	3.081 (* 6.77%)
6	5.278	3.523	5.417 (* 6.91%)	3.562 (* 4.08%)
7	6.271	4.016	6.369 (* 4.17%)	4.042 (* 2.40%)
8	7.264	4.510	7.327 (* 2.46%)	4.527 (* 1.37%)
9	8.257	5.008	8.296 (* 1.42%)	5.018 (* 0.77%)

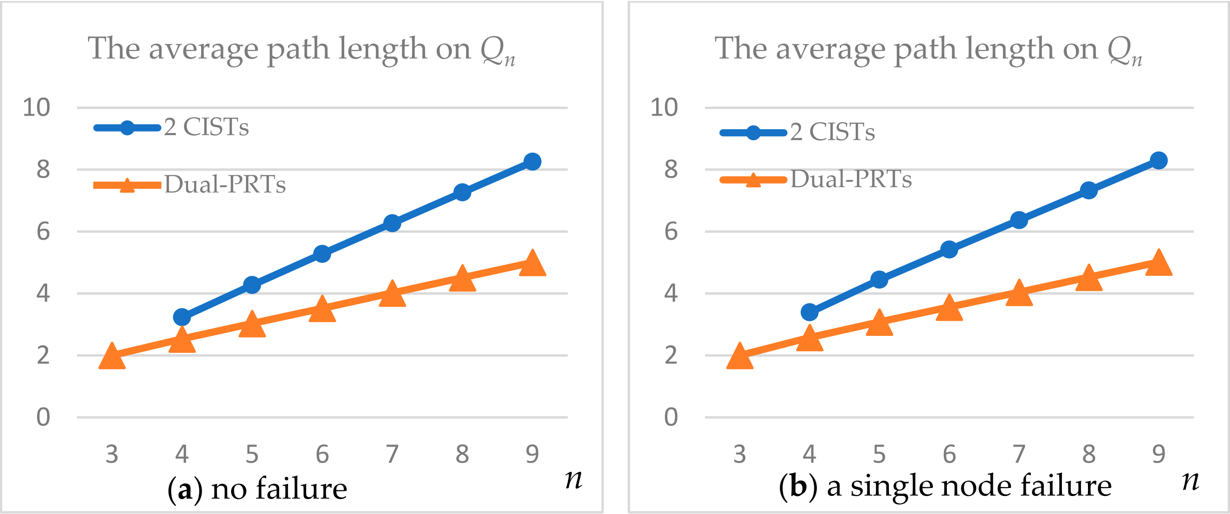
* The ratio of invoking SNH.

Table 3. Simulation results: the average path length on LTQ_n when the routing is with two scenarios: (1) no failure; and (2) a single node failure.

	No Failure		**A Single Node Failure**	
n	**2 CISTs**	**Dual-PRT^{d}s**	**2 CISTs**	**Dual-PRT^{d}s**
3		2.285		2.237 (* 21.41%)
4	2.932	2.600	3.027 (* 13.83%)	2.637 (* 11.44%)
5	3.869	3.290	3.983 (* 9.53%)	3.325 (* 7.64%)
6	4.826	3.870	4.914 (* 6.16%)	3.902 (* 4.59%)
7	5.794	4.411	5.853 (* 3.78%)	4.434 (* 2.67%)
8	6.774	4.926	6.811 (* 2.26%)	4.941 (* 1.55%)
9	7.766	5.432	7.788 (* 1.34%)	5.442 (* 0.85%)

* The ratio of invoking SNH.

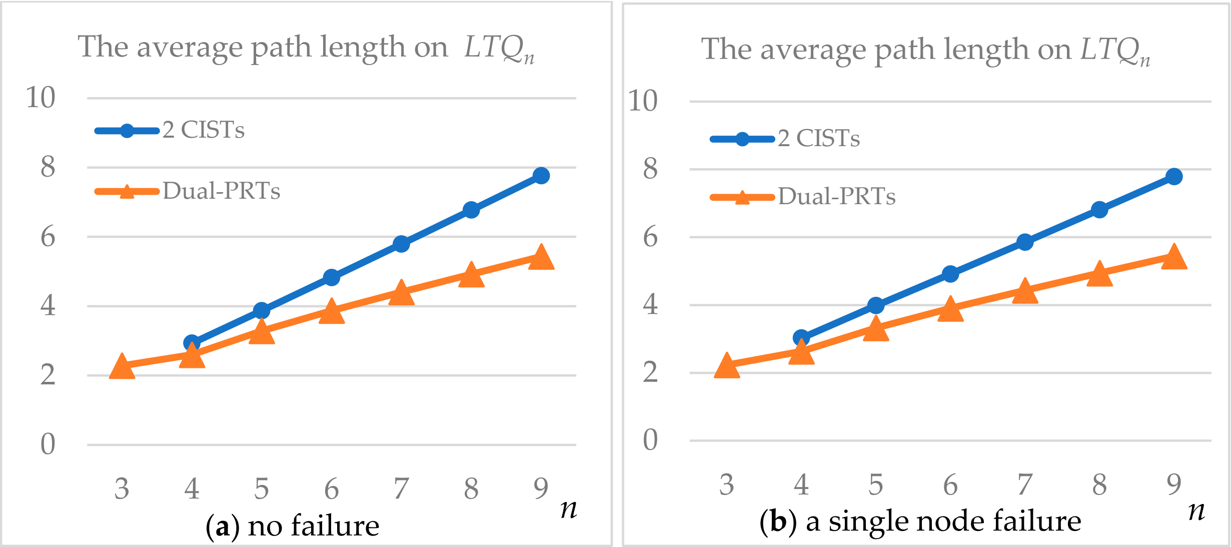
According to Tables 2 and 3, we have Figures 11 and 12. There are no two CISTs in Q_3 by Lemma 1, which is why the blue line (representing 2 CISTs) in Figure 11 starts with count $n = 4$. For the same reason, the blue line in Figure 12 also starts with count $n = 4$. Obviously, the average path lengths of protection routing by dual-PRT^{d}s are smaller than the ones by two CISTs, no matter the scenario.



(**a**) no failure

(**b**) a single node failure

Figure 11. The average path lengths of protection routing on Q_n while $3 \leq n \leq 9$.



(**a**) no failure

(**b**) a single node failure

Figure 12. The average path lengths of protection routing on LTQ_n while $3 \leq n \leq 9$.

5. Conclusions

In our previous research on using CISTs to construct the protection routing, we found that it is not easy to find k-CISTs on some graph classes even when k is equal to 2. We tried to find trees that are less restrictive but can still be used to configure the protection routing. Then, we proposed dual-PRT^{d}s in this paper. As mentioned earlier, there are no two CISTs on $K_{3,3}$, Q_3, and LTQ_3. However, there are dual-PRT^{d}s on them, which can be used to construct the protection routing. These results improve on previous research. Additionally, as shown in the performance evaluation of protection routing through simulation results, the average path length of protection routing configured by dual-PRT^{d}s is shorter than that of two CISTs. The dual-PRT^{d}s we proposed are modified from k-CISTs, and k-CISTs have dozens of research results on different network architectures. For future research, it would be an interesting question to determine the existence of dual-PRT^{d}s in other graph classes and practical implementation on a general network.

Supplementary Materials: The full automorphism groups of Q_n and LTQ_n, simulation description, and detailed results can be viewed at the following website: https://github.com/kjpai/dprt1/blob/main/dprt1content.pdf.

Funding: This research was funded by MOST through the Ministry of Science and Technology, Taiwan, under Grant MOST 111-2221-E-131-012.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Kwong, K.-W.; Gao, L.; Guérin, R.; Zhang, Z.-L. On the feasibility and efficacy of protection routing in IP networks. *IEEE/ACM Trans. Netw.* **2011**, *19*, 1543–1556. [CrossRef]
2. Hasunuma, T. Completely independent spanning trees in the underlying graph of a line digraph. *Discrete Math.* **2001**, *234*, 149–157. [CrossRef]
3. Tapolcai, J. Sufficient conditions for protection routing in IP networks. *Optim. Lett.* **2013**, *7*, 723–730. [CrossRef]
4. Hasunuma, T. Completely independent spanning trees in maximal planar graphs. In *WG 2002. LNCS*; Goos, G., Hartmanis, J., van Leeuwen, J., Kučera, L., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2573, pp. 235–245.
5. Hasunuma, T.; Morisaka, C. Completely independent spanning trees in torus networks. *Networks* **2012**, *60*, 59–69. [CrossRef]
6. Cheng, B.; Wang, D.; Fan, J. Constructing completely independent spanning trees in crossed cubes. *Discret. Appl. Math.* **2017**, *219*, 100–109. [CrossRef]
7. Darties, B.; Gastineau, N.; Togni, O. Completely independent spanning trees in some regular graphs. *Discret. Appl. Math.* **2017**, *217*, 163–174. [CrossRef]
8. Pai, K.-J.; Tang, S.-M.; Chang, J.-M.; Yang, J.-S. Completely independent spanning trees on complete graphs, complete bipartite graphs and complete tripartite graphs. In *Advances in Intelligent Systems and Applications-Volume 1, Proceedings of the International Computer Symposium ICS 2012, Hualien, Taiwan, 12–14 December 2012*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 20, pp. 107–113.
9. Pai, K.-J.; Chang, J.-M. Constructing two completely independent spanning trees in hypercube-variant networks. *Theor. Comput. Sci.* **2016**, *652*, 28–37. [CrossRef]
10. Pai, K.J.; Yang, J.S.; Chen, G.Y.; Chang, J.M. Configuring Protection Routing via Completely Independent Spanning Trees in Dense Gaussian On-Chip Networks. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 932–946. [CrossRef]
11. Samavi, S.; Khadivi, P. Fault-tolerant routing in hypercube networks by avoiding faulty nodes. *arXiv* **2019**, arXiv:1905.03086.
12. Bossard, A.; Kaneko, K. Cluster-Fault Tolerant Routing in a Torus. *Sensors* **2020**, *20*, 3286. [CrossRef]
13. Shu, C.; Wang, Y.; Fan, J.; Zhang, H. Fault-Tolerant Routing of Generalized Hypercubes under 3-Component Connectivity. In Proceedings of the 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking, New York, NY, USA, 30 September–3 October 2021; pp. 1320–1327.
14. Romanov, A.Y.; Myachin, N.M.; Lezhnev, E.V.; Ivannikov, A.D.; El-Mesady, A. Ring-Split: Deadlock-Free Routing Algorithm for Circulant Networks-on-Chip. *Micromachines* **2023**, *14*, 141. [CrossRef]
15. Leighton, F.T. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*; Morgan Kaufmann: San Mateo, CA, USA, 1992.
16. Yang, X.; Evans, D.J.; Megson, G.M. The locally twisted cubes. *Int. J. Comput. Math.* **2005**, *82*, 401–413. [CrossRef]
17. Chang, X.; Ma, J.; Yang, D.-W. Symmetric property and reliability of locally twisted cubes. *Discret. Appl. Math.* **2021**, *288*, 257–269. [CrossRef]
18. Simulation Results for Evaluating the Performance of the Protection Routings on Qn and LTQn. Available online: https://github.com/kjpai/dprt1/blob/main/dprt1content.pdf (accessed on 7 July 2023).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.