Discrete Applied Mathematics 160 (2012) 2006–2014

Contents lists available at SciVerse ScienceDirect

Discrete Applied Mathematics

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

One-to-many node-disjoint paths of hyper-star networks

László Lipták [a,*], Eddie Cheng [a], Jong-Seok Kim [b], Sung Won Kim [b]

[a] *Department of Mathematics and Statistics, Oakland University, Rochester, MI 48309, United States*

[b] *Department of Information and Communication Engineering, Yeungnam University, Gyeongsan, Gyeongbuk, 712-749, Republic of Korea*

ARTICLE INFO

Article history:
Received 11 May 2011
Received in revised form 13 December 2011
Accepted 9 April 2012
Available online 4 May 2012

Keywords:
Interconnection network
Hyper-star
One-to-many node-disjoint path
Algorithms

ABSTRACT

In practice, it is important to construct node-disjoint paths in networks, because they can be used to increase the transmission rate and enhance the transmission reliability. The hyper-star networks $\mathrm{HS}(2n, n)$ were introduced to be a competitive model for both the hypercubes and the star graphs. In this paper, one-to-many node-disjoint paths are constructed between a fixed node and n other nodes of $\mathrm{HS}(2n, n)$ such that each of these paths has length at most 4 more than the shortest path to that node. Moreover, their maximum length is not greater than the diameter + 2.

1. Introduction

Advances in hardware technology, especially very-large-scale integration circuit technology, have made it possible to build a large-scale multiprocessor system that contains thousands or even tens of thousands of processors. One crucial step in designing a large-scale multiprocessor system is to determine the topology of the interconnection network (network for short), because the system performance is significantly affected by the network topology. In recent decades, a number of networks have been proposed in the literature [10,16]. A network is conveniently represented by a graph whose nodes represent the processors of the network and whose edges represent the communication links of the network. Throughout this paper, we use network and graph, processor and node, and link and edge, interchangeably.

Let $G = (V, E)$ be a connected graph, where V and E represent the node set and edge set of G, respectively. The *degree* of a node in G is the number of edges incident with it. If all nodes have the same degree d, then G is called *regular*. The *distance* between two nodes u and v, denoted by $\mathrm{dist}(u, v)$, is the length of a shortest path between u and v. The *diameter* of G is the maximum distance between any two nodes of G. The *node connectivity* of G is the minimal number of nodes in G whose removal can cause G to become disconnected or trivial.

One of the most efficient interconnection networks is the hypercube [11]. Another family of regular graphs, the star graphs [1], has been extensively studied. The hyper-star graphs $\mathrm{HS}(m, n)$ were introduced by Lee et al. [15] and Kim et al. [13] to become a new type of interconnection networks for competing with both the hypercubes and the star graphs. The hyper-star network is a regular network only when $m = 2n$. A result by Lee et al. [15] also showed that hyper-star graphs gave lower network cost (measured by the product of degree and diameter) than hypercubes, folded hypercubes, and other variants. Later on, stronger structural properties and some embedding schemes for hyper-star graphs were provided respectively in [4,5] and in [12].

In practice, it is important to construct node-disjoint paths in networks, because they can be used to increase the transmission rate and enhance the transmission reliability. Besides that, node-disjoint paths have applications in multipath

* Corresponding author. Tel.: +1 248 370 4054; fax: +1 248 370 4184.
E-mail address: liptak@oakland.edu (L. Lipták).

0166-218X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2012.04.006

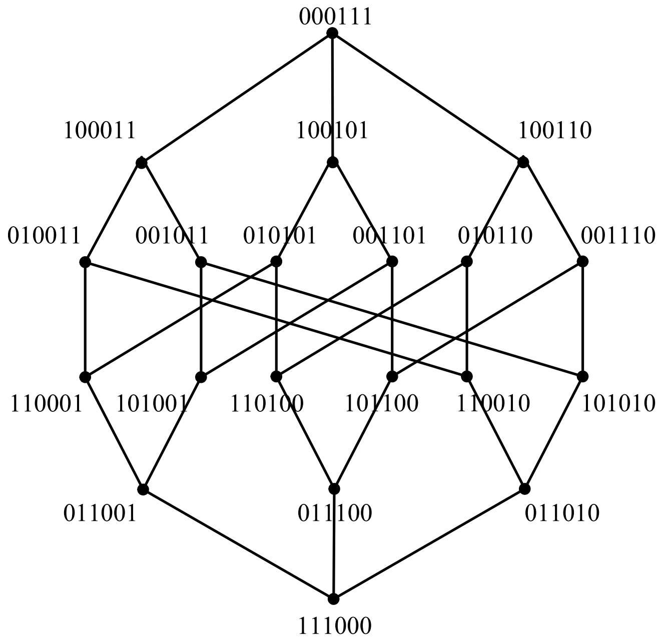

Fig. 1. HS(6, 3).

routing such as Rabin's information dispersal algorithm [18], fault tolerance [6,7], and communication protocols [10]. There are two paradigms for the study of node-disjoint paths in interconnection networks: the one-to-one routing that constructs the maximum number of node-disjoint paths in the network between two given nodes, and the one-to-many routing that constructs internally node-disjoint paths in the network from a given node to each of the nodes in a given set. One-to-one node-disjoint paths in a variety of networks can be found in [7,9,10,13,14,17,19,20], while one-to-many node-disjoint paths were examined for the hypercube in [2,8].

One-to-one node-disjoint paths in the hyper-star network HS$(2n, n)$ have been studied by Kim et al. [13]. In this paper, we propose an algorithm to construct one-to-many node-disjoint paths in HS$(2n, n)$. Their maximum length will not be greater than $2n + 1$, which is at most 2 bigger than $2n - 1$, the diameter of HS$(2n, n)$.

The remainder of this paper is organized as follows. In Section 2, we introduce HS$(2n, n)$ and some of its useful properties, and in Section 3 we propose our algorithm to construct one-to-many internally node-disjoint paths in HS$(2n, n)$ whose maximum length will not be greater than $2n + 1$. Finally, a conclusion is given in Section 4.

2. Preliminaries

The *hyper-star graph* HS(m, n) is an undirected graph consisting of $\binom{m}{n}$ nodes, where each node is represented by a binary string of m bits $b_1 b_2 \cdots b_i \cdots b_m$ such that the number of bits equal to 1 is n (i.e., $|\{i : 1 \le i \le m, b_i = 1\}| = n$). Two nodes are adjacent if and only if one can be obtained from the other by exchanging its first bit with a different bit (1 with 0, or 0 with 1) in another position. An edge connecting two nodes u and v is called an *i-edge* if it results from switching the first bit of u with its ith bit. For example, in HS(6, 3), node 011001 is adjacent to node 110001 by a 3-edge. Clearly, every node in HS(m, n) has degree n or $m - n$, and HS(m, n) is regular if and only if $m = 2n$. Fig. 1 shows HS(6, 3).

Note that HS$(2n, n)$ is isomorphic to the middle cube, the subgraph spanned by the nodes containing n or $n - 1$ 1s in the hypercube H_{2n-1} (delete the first bit of each node to get an isomorphism). Incidentally, many relatively easy results for the hypercube are difficult to prove for the hyper-star. For example, a famous conjecture is that HS$(2n, n)$ is Hamiltonian, which is called the revolving door conjecture.

Let dist(u, v) be the distance from $u = u_1 u_2 \cdots u_{2n}$ to $v = v_1 v_2 \cdots v_{2n}$. If R is the bit string obtained by applying the bitwise Exclusive-OR operation to them, thus $R = r_1 r_2 \cdots r_{2n}$, where $r_i = u_i \oplus v_i$, then dist$(u, v) = \sum_{i=2}^{2n} r_i$. For a node u, we denote by $[k_1, k_2, \ldots, k_t]$ the path obtained by starting from u, going to its neighbor using a k_1-edge, going to its neighbor using a k_2-edge, etc., assuming that such an edge is actually present in the graph. For example, for $u = 000111$, the sequence [4, 2, 6] represents the path 000111–100011–010011–110010 in HS(6, 3), but the sequence [4, 5, 6] does not represents a path (or walk), since the second move is not allowed (it switches two 1s). Clearly, every path can be represented in such a way, though not every sequence represents a path (or even a walk) for a given starting node. Notice that if $k_1, k_2, \ldots, k_t$ are all different, and $[k_1, k_2, \ldots, k_t]$ represents a path from u to v, then we can permute the k_is with even subscripts and we can permute the k_is with odd subscripts and still get a path from u to v of the same length (other permutations do not correspond to paths). Thus the same applies to shortest paths. Two paths are *internally node disjoint* if any common node on the paths is an endpoint of both paths. We will use 0^n1^n to represent the node $\overbrace{0\cdots0}^{n}\overbrace{1\cdots1}^{n}$ in $\mathrm{HS}(2n, n)$.

We will need the following easy lemmas about the cycles in $\mathrm{HS}(2n, n)$.

Lemma 1 (*[5]*). *The length of a shortest cycle in* $\mathrm{HS}(2n, n)$ *is* 6.

Lemma 2. *Let u and v be two nodes in* $\mathrm{HS}(2n, n)$, *and let P and Q be paths with length $\rho \geq 3$ from u to v, represented respectively by the sequences $[k_1, k_2, \ldots, k_\rho]$ and $[m_1, m_2, \ldots, m_\rho]$, where $k_1, k_2, \ldots, k_\rho$ are all different. There are no common internal nodes on P and Q if and only if, for every $1 \leq i < \rho$, we have $\{k_1, k_2, \ldots, k_i\} \neq \{m_1, m_2, \ldots, m_i\}$.*

Proof. Note that, since $k_1, \ldots, k_\rho$ are all different, as we follow path P from u to v, these bits are flipped exactly once, so, in order for Q to end at the same node, we must have $\{k_1, \ldots, k_\rho\} = \{m_1, \ldots, m_\rho\}$. First, assume that there are no common internal nodes on P and Q. Then, for $1 \leq i < \rho$, the subpaths represented by $[k_1, k_2, \ldots, k_i]$ and $[m_1, m_2, \ldots, m_i]$ end at different nodes, so we must have $\{k_1, k_2, \ldots, k_i\} \neq \{m_1, m_2, \ldots, m_i\}$.

For the other direction, assume that $\{k_1, k_2, \ldots, k_i\} \neq \{m_1, m_2, \ldots, m_i\}$ for every $1 \leq i < \rho$. Assume that there is a common internal node w on the two paths, and that the subpaths to w are respectively represented by $[k_1, k_2, \ldots, k_j]$ and $[m_1, m_2, \ldots, m_p]$, where j and p are both less than ρ. Since along the path represented by $[k_1, k_2, \ldots, k_j]$ each corresponding bit is flipped exactly once, and $m_1, m_2, \ldots, m_p$ are all different, we must have $j = p$ and $\{k_1, k_2, \ldots, k_j\} = \{m_1, m_2, \ldots, m_j\}$. $\square$

This lemma can be easily generalized for paths possibly ending at different nodes as follows.

Lemma 3. *Let u, v, w be nodes in* $\mathrm{HS}(2n, n)$ *with $v \neq w$, and let P and Q be paths from u to respectively v and w. Let P and Q be represented respectively by the sequences $[k_1, k_2, \ldots, k_\rho]$ and $[m_1, m_2, \ldots, m_\sigma]$, where $\rho, \sigma \geq 3$, and assume that $k_1, k_2, \ldots, k_\rho$ are all different and $m_1, m_2, \ldots, m_\sigma$ are all different. There are no common internal nodes on P and Q if and only if, for every $1 \leq i \leq \min\{\rho, \sigma\}$, we have $\{k_1, k_2, \ldots, k_i\} \neq \{m_1, m_2, \ldots, m_i\}$.*

Proof. First, assume that there are no common internal nodes on P and Q. Then, for $1 \leq i \leq \min\{\rho, \sigma\}$, the subpaths represented by $[k_1, k_2, \ldots, k_i]$ and $[m_1, m_2, \ldots, m_i]$ end at different nodes (when $i = \rho = \sigma$, this follows from $v \neq w$). Since along each subpath each bit corresponding to an element of these sequences is flipped exactly once, we must have $\{k_1, k_2, \ldots, k_i\} \neq \{m_1, m_2, \ldots, m_i\}$.

For the other direction, assume that $\{k_1, k_2, \ldots, k_i\} \neq \{m_1, m_2, \ldots, m_i\}$ for every $1 \leq i \leq \min\{\rho, \sigma\}$. By contradiction, assume that there is a common node $z \neq u$ on the two paths, and that the subpaths to z are respectively represented by $[k_1, k_2, \ldots, k_j]$ and $[m_1, m_2, \ldots, m_p]$, where $j \leq \rho$ and $p \leq \sigma$. Since along the path represented by $[k_1, k_2, \ldots, k_j]$ each corresponding bit is flipped exactly once, and $m_1, m_2, \ldots, m_p$ are also all different, we must have $j = p \leq \min\{\rho, \sigma\}$ and $\{k_1, k_2, \ldots, k_j\} = \{m_1, m_2, \ldots, m_j\}$, which is a contradiction. $\square$

Remark. Lemma 3 can be generalized further for the case when the paths may contain repeated elements by noticing the following: two paths from u represented by $[k_1, k_2, \ldots, k_\rho]$ and $[m_1, m_2, \ldots, m_\sigma]$ will end up at the same node if and only if, for every number i, the number of occurrences of i in the two sequences have the same parity.

3. One-to-many node-disjoint paths in HS($2n$, n)

In this section, we present our algorithm to find node-disjoint paths in $\mathrm{HS}(2n, n)$. We first introduce some notation that will help us describe the paths. $CR_x(S)$ will denote the sequence obtained from the sequence S by rotating its elements to the left x times. For example, if $S = [1, 2, \ldots, n]$, then $CR_0(S) = [1, 2, \ldots, n]$, and $CR_3(S) = [4, 5, \ldots, n, 1, 2, 3]$. Assume that P is a path connecting two nodes u and v in the hyper-star $\mathrm{HS}(2n, n)$ and that the sequence S represents P starting from u. Pick the numbers in the odd positions from S to form $S_1 = [a_1, a_2, \ldots, a_p]$, and pick the ones in the even positions to form $S_2 = [b_1, b_2, \ldots, b_q]$. Given sequences S_1 and S_2 with $p = q$ or $p = q + 1$, $S_1 \otimes S_2$ will represent the new sequence obtained by alternately picking elements of S_1 and S_2 (finishing with the last element of S_1 if $p = q + 1$). For instance, if $S_1 = [5, 6, 7]$ and $S_2 = [2, 3, 4]$, then $S_1 \otimes S_2 = [5, 2, 6, 3, 7, 4]$. Clearly, using $x = 0, 1, \ldots, p - 1$, we can get p different paths of the form $CR_x(S_1) \otimes CR_x(S_2)$, and they will be internally node disjoint by Lemma 2.

Since $\mathrm{HS}(2n, n)$ is node symmetric (see [4]), we may fix node $u = 0^n1^n$, and let $v = b_1b_2\cdots b_i\cdots b_{2n}$ be another node of $\mathrm{HS}(2n, n)$. Let the result of applying the bitwise Exclusive-OR function on these two nodes be the bitstring $R = r_1r_2\cdots r_i\cdots r_{2n}$ (so $r_i = u_i \oplus b_i$), and let $\mathrm{dist}(u, v) = t$. Let the set R^1 consist of bit positions i such that $r_i = 1$ and $2 \leq i \leq 2n$, so $|R^1| = t$. We divide the elements of R^1 into the following two sequences: bit positions up to n are put into H_1, bit positions that are at least $n+1$ are put into H_2, both in an increasing order (so H_1 and H_2 depend on v). Thus, if $R^1 = \{i_1, i_2, \ldots, i_t\}$ such that $i_1 < i_2 < \cdots < i_g \leq n < i_{g+1} < \cdots < i_t$, then $H_1 = [i_1, i_2, \ldots, i_g]$ and $H_2 = [i_{g+1}, i_{g+2}, \ldots, i_t]$. It is easy to see that $g = \frac{t-1}{2}$ if t is odd, while $g = \frac{t}{2}$ if t is even. Similarly, let the set R^0 consist of bit positions i such that $r_i = 0$ and $2 \leq i \leq 2n$. We divide the elements of R^0 into two sequences H_3 and H_4 the same way: bit positions up to n go into H_3, and the rest go into H_4, both in an increasing order. Thus if $R^0 = \{i_1, i_2, \ldots, i_{t'}\}$ such that $i_1 < i_2 < \cdots < i_f \leq n < i_{f+1} < \cdots < i_{t'}$ (where

$t' = 2n - 1 - t$), then $H_3 = [i_1, i_2, \ldots, i_f]$ and $H_4 = [i_{f+1}, i_{f+2}, \ldots, i_{t'}]$. Again, it is easy to see that $f = \frac{t'-1}{2}$ if t' is odd, while $f = \frac{t'}{2}$ if t' is even. Using the above notation, we can find $\lceil \frac{t}{2} \rceil$ paths of the form $CR_x(H_2) \otimes CR_x(H_1)$ from u to v. For example, with $n = 4$ and $v = 10110100$, we get $R^1 = \{3, 4, 5, 7, 8\}$, so $H_1 = [3, 4]$ and $H_2 = [5, 7, 8]$, while $R^0 = \{2, 6\}$, so $H_3 = [2]$ and $H_4 = [6]$. We can get $\lceil \frac{5}{2} \rceil = 3$ internally node-disjoint paths from u to v of the form $CR_x(H_2) \otimes CR_x(H_1)$ by using $x = 0, 1, 2$: [5, 3, 7, 4, 8], [7, 4, 8, 3, 5], and [8, 3, 5, 4, 7].

Since HS$(2n, n)$ is n-connected (in fact, it has stronger properties; see [4,5,3]), we can construct n one-to-many node-disjoint paths. Let the destination nodes be $v_1, \ldots, v_n$; order them such that their distances are non-decreasing: $\mathrm{dist}(u, v_1) \le \mathrm{dist}(u, v_2) \le \cdots \le \mathrm{dist}(u, v_n)$. We want to find a path P_k from $u = 0^n 1^n$ to v_k for each k, $k = 1, \ldots, n$. Let the result of applying the bitwise Exclusive-OR function on the two nodes u and v_k be the bitstring $R_k = r_{k,1} r_{k,2} \cdots r_{k,2n}$ for $k = 1, \ldots, n$.

We define two matrices, M_1 and M_2, as follows. The rows of M_2 will be composed of the bits of the bitstrings $R_k^{M_2} = r_{k,n+1} r_{k,n+2} \cdots r_{k,2n}$ for $1 \le k \le n$, while the rows of M_1 will be composed of the bits of the corresponding bitstrings $R_k^{M_1} = r_{k,2} r_{k,3} \cdots r_{k,n}$, $1 \le k \le n$ in the same order:

$$M_1 = \begin{pmatrix} r_{1,2} & r_{1,3} & \cdots & r_{1,n} \\ r_{2,2} & r_{2,3} & \cdots & r_{2,n} \\ \vdots & & \ddots & \\ r_{n,2} & r_{n,3} & \cdots & r_{n,n} \end{pmatrix} \qquad M_2 = \begin{pmatrix} r_{1,n+1} & r_{1,n+2} & \cdots & r_{1,2n} \\ r_{2,n+1} & r_{2,n+2} & \cdots & r_{2,2n} \\ \vdots & & \ddots & \\ r_{n,n+1} & r_{n,n+2} & \cdots & r_{n,2n} \end{pmatrix}$$

Next, we choose the first edge on each of the paths. Clearly, each of these edges must be an i-edge for some $i > n$, because $u = 0^n 1^n$. First, select as many 1s from M_2 as possible with the restriction that we can select at most one 1 from each row and from each column. This can be done by finding a maximum matching in an auxiliary bipartite graph whose nodes correspond to the rows and columns of M_2, and there is an edge between a row and a column if the corresponding entry in M_2 is 1. Then modify this selection so that the 1s are chosen for the nodes closest to u as follows. For every column in which a 1 has been selected, check if it has a 1 above it in a row in which no 1 has been selected. If this happens, switch the later 1 to the earlier 1. Repeat until no such 1 is found. Then, in each row where no bit has been selected yet, select the bit 0 in the first column in which no bit has been selected yet. Continue this way until exactly one bit (either 0 or 1) is selected in each row and each column. For each $k = 1, \ldots, n$, if the bit $r_{k,j}$ was selected, it will be indicated by $\widetilde{r_{k,j}}$, and then we choose the first edge on P_k to be a j-edge. Since a maximum matching can be found in $O(NM)$ time if the graph has N nodes and M edges, this preliminary step can be done in $O(n^3)$ time.

Example 1. (Here and in later examples we will put a vertical bar in the middle of each string to make it easier to identify $H_1, \ldots, H_4$.) Let $n = 4$, $u = 0000|1111$, and $v_1 = 1010|0011$, $v_2 = 0110|0011$, $v_3 = 0110|0101$, and $v_4 = 1110|0001$. So $R_1 = 1010|1100$, $R_2 = 0110|1100$, $R_3 = 0110|1010$, and $R_4 = 1110|1110$. Therefore

$$M_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}, \qquad M_2 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

Assume that we first pick the maximum possible number of 1s as follows: $\widetilde{r_{2,5}}, \widetilde{r_{3,7}}, \widetilde{r_{4,6}}$ (note that columns of M_2 are labeled with $5, \ldots, 8$), giving

$$M_2 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ \widetilde{1} & 1 & 0 & 0 \\ 1 & 0 & \widetilde{1} & 0 \\ 1 & \widetilde{1} & 1 & 0 \end{pmatrix}.$$

There is a 1 above $\widetilde{r_{2,5}}$ in the first row, so we switch it to $\widetilde{r_{1,5}}$. After that there is a 1 above $\widetilde{r_{4,6}}$ in the second row (there is a 1 in the first row as well, but we already picked a 1 from that row), so we switch it to $\widetilde{r_{2,6}}$. This gives

$$M_2 = \begin{pmatrix} \widetilde{1} & 1 & 0 & 0 \\ 1 & \widetilde{1} & 0 & 0 \\ 1 & 0 & \widetilde{1} & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

Finally, choose a 0 in each row where no bit has been selected: $\widetilde{r_{4,8}}$, so

$$M_2 = \begin{pmatrix} \widetilde{1} & 1 & 0 & 0 \\ 1 & \widetilde{1} & 0 & 0 \\ 1 & 0 & \widetilde{1} & 0 \\ 1 & 1 & 1 & \widetilde{0} \end{pmatrix}.$$

Now, for each $k = 1, \dots, n$, we choose the path P_k from u to v_k. The idea is to choose some preliminary paths in Stage 1, then resolve any conflicts by using Stage 2 as follows.

Path Algorithm, Stage 1

Let $H_1^k, \dots, H_4^k$ be the sets corresponding to v_k, let the first edge chosen to be on P_k be a j-edge (so $\widetilde{r_{k,j}}$), and let x be the number of left cyclic rotations needed to move j to be the first element in $CR_x(H_2^k)$ when $\widetilde{r_{k,j}} = 1$.

Case 1. If $\widetilde{r_{k,j}} = 1$, then let $P_k = CR_x(H_2^k) \otimes H_1^k$.

Case 2. If $\widetilde{r_{k,j}} = 0$, and the distance from u to v_k is even, then let $P_k = [j, H_1^k \otimes H_2^k, j]$.

Case 3. If $\widetilde{r_{k,j}} = 0$, and the distance from u to v_k is odd, then let $P_k = [j, i_k, H_2^k \otimes H_1^k, j, i_k]$, where i_k is the first element in H_3^k.

Note that in Case 3 H_3^k cannot be empty, since that would imply that $v_k = 1^n0^n$. But then $H_2^k = \{n+1, \dots, 2n\}$, so we must have picked a bit that is 1, so v_k should fall under Case 1. In addition, the distance from u to v_k in this case is at most $2n-3$ (must be odd and strictly less than $2n-1$), so the maximum path length after Stage 1 is $2n+1$. So none of these paths have length larger than diameter + 2 of HS$(2n, n)$. Clearly Stage 1 can be done in $O(n^2)$ time (the total length of all paths).

Stage 1 is not guaranteed to find internally node-disjoint paths, so we need to check whether there is a conflict, and modify at least one of the paths when there is one. This is achieved in Stage 2 of the algorithm.

Path Algorithm, Stage 2

Step 1. Check if there are two paths $P_\beta = [p_1, p_2, \dots, p_\gamma]$ and $Q_\delta = [q_1, q_2, \dots, q_\eta]$ with $\beta \neq \delta$ such that $\widetilde{r_{\beta,p_1}} = \widetilde{r_{\delta,q_1}} = 1$ and the two paths have a common node which is not the endpoint of both paths. If there are such paths and $H_2^\beta \neq H_2^\delta$, then switch the chosen bits ($\widetilde{r_{\beta,q_1}} = \widetilde{r_{\delta,p_1}} = 1$) and redefine both paths according to Stage 1 of the algorithm ($P_k = CR_x(H_2^k) \otimes H_1^k$). If $H_2^\beta = H_2^\delta$, then consider all nodes with this same H_2 (and chosen bit 1), and redefine the paths as follows. For nodes with an even distance to u, permute the corresponding sets H_1^k in the paths such that, if we delete their last elements, then they become all different, and they will be different from the sets H_1 corresponding to nodes with this same H_2 with an odd distance from u.

Repeat Step 1 until no such pairs are found.

Step 2. Check if there are two paths $P_\beta = [p_1, p_2, \dots, p_\gamma]$ and $Q_\delta = [q_1, q_2, \dots, q_\eta]$ with $\beta \neq \delta$ and $H_2^\beta = H_2^\delta$ such that $\widetilde{r_{\beta,p_1}} = \widetilde{r_{\delta,q_1}} = 0$ and the two paths have a common internal node. If there are such paths, then consider all nodes with this same H_2 (and chosen bit 0), and redefine the paths as follows. For nodes with an odd distance from u, choose i_k from H_3^k such that the sets $H_1^k \cup \{i_k\}$ are all different, and they are all different from the sets H_1 corresponding to nodes with the same H_2 at an even distance from u. Redefine these paths according to Case 3 of Stage 1 using the new i_k.

Repeat Step 2 until no such pairs are found.

Before we show that the algorithm will terminate in polynomial time and finish with internally node-disjoint paths, we give a few examples to illustrate Stage 2.

Example 2. Let $n = 4$ and $v_1 = 0110|0101$, $v_2 = 0110|0011$, $v_3 = 1110|0001$. Then $H_1^1 = H_1^2 = H_1^3 = \{2, 3\}$, $H_2^1 = \{5, 7\}$, $H_2^2 = \{5, 6\}$, $H_2^3 = \{5, 6, 7\}$, and let M_2 with the chosen bits be

$$M_2 = \begin{pmatrix} \tilde{1} & 0 & 1 & 0 \\ 1 & \tilde{1} & 0 & 0 \\ 1 & 1 & \tilde{1} & 0 \end{pmatrix}.$$

Stage 1 will assign the following paths: $P_1 = [5, 2, 7, 3]$, $P_2 = [6, 2, 5, 3]$, $P_3 = [7, 2, 5, 3, 6]$. Step 1 of Stage 2 will find that the third node is a common internal node on P_1 and P_3 since $\{5, 2, 7\} = \{7, 2, 5\}$. Their corresponding H_2s are different, so we switch their chosen bits to get

$$M_2 = \begin{pmatrix} 1 & 0 & \tilde{1} & 0 \\ 1 & \tilde{1} & 0 & 0 \\ \tilde{1} & 1 & 1 & 0 \end{pmatrix},$$

and the paths are redefined as $P_1 = [7, 2, 5, 3]$, $P_2 = [6, 2, 5, 3]$, and $P_3 = [5, 2, 6, 3, 7]$. Now the third node is a common internal node on P_2 and P_3 since $\{6, 2, 5\} = \{5, 2, 6\}$. Their corresponding H_2s are different, so we switch their chosen bits to get

$$M_2 = \begin{pmatrix} 1 & 0 & \tilde{1} & 0 \\ \tilde{1} & 1 & 0 & 0 \\ 1 & \tilde{1} & 1 & 0 \end{pmatrix},$$

and the paths are redefined as $P_1 = [7, 2, 5, 3]$, $P_2 = [5, 2, 6, 3]$, and $P_3 = [6, 2, 7, 3, 5]$. Now the paths are internally node-disjoint, and the algorithm terminates.

	p_1		p_{k-2}	p_k				p_j	
H_2^{β}:	34	37	41	44	48	25	29	31	33
H_2^{δ}:	44	48	25	29	31	34	37	41	42
	q_1				q_{m-2}	q_m		q_j	

Fig. 2. Example for P_β and Q_δ with a common node in HS(48, 24).

Example 3. Let $n = 5$ and $v_1 = 11100|00011$, $v_2 = 01110|00011$, $v_3 = 01101|00011$. Then $H_1^1 = \{2, 3\}$, $H_1^2 = \{2, 3, 4\}$, $H_1^3 = \{2, 3, 5\}$, $H_2^1 = H_2^2 = H_2^3 = \{6, 7, 8\}$, and let M_2 with the chosen bits be

$$M_2 = \begin{pmatrix} \tilde{1} & 1 & 1 & 0 & 0 \\ 1 & \tilde{1} & 1 & 0 & 0 \\ 1 & 1 & \tilde{1} & 0 & 0 \end{pmatrix}.$$

Stage 1 will assign the following paths: $P_1 = [6, 2, 7, 3, 8]$, $P_2 = [7, 2, 8, 3, 6, 4]$, $P_3 = [8, 2, 6, 3, 7, 5]$. Step 1 of Stage 2 will find that these paths have a common internal node (the last node of P_1 is an internal node of P_2 and P_3), and these nodes have the same H_2. We permute the H_1s of v_2 and v_3 (these nodes have even distance from u) so that excluding their last elements will result in different sets which are also different from H_1^1. One possibility is to use $CR_2(H_1^2) = [4, 2, 3]$ and $CR_1(H_1^3) = [3, 5, 2]$, since $\{2, 3\}$, $\{4, 2\}$, and $\{3, 5\}$ are all different (we explain how to find these permutations in Theorem 1). Then P_2 and P_3 are redefined to get $P_1 = [6, 2, 7, 3, 8]$, $P_2 = [7, 4, 8, 2, 6, 3]$, $P_3 = [8, 3, 6, 5, 7, 2]$, making them internally node disjoint, and the algorithm terminates.

Example 4. Let $n = 6$ and $v_1 = 010000|011111$, $v_2 = 001000|101111$, $v_3 = 000100|110111$, $v_4 = 111000|000111$, $v_5 = 101100|000111$, $v_6 = 011100|000111$. Then $H_1^1 = \{2\}$, $H_1^2 = \{3\}$, $H_1^3 = \{4\}$, $H_1^4 = \{2, 3\}$, $H_1^5 = \{3, 4\}$, $H_1^6 = \{2, 3, 4\}$, $H_2^1 = \{7\}$, $H_2^2 = \{8\}$, $H_2^3 = \{9\}$, $H_2^4 = H_2^5 = H_2^6 = \{7, 8, 9\}$, and let M_2 with the chosen bits be

$$M_2 = \begin{pmatrix} \tilde{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \tilde{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \tilde{1} & 0 & 0 & 0 \\ 1 & 1 & 1 & \tilde{0} & 0 & 0 \\ 1 & 1 & 1 & 0 & \tilde{0} & 0 \\ 1 & 1 & 1 & 0 & 0 & \tilde{0} \end{pmatrix}.$$

Stage 1 will assign the following paths using $i_4 = 4$ and $i_5 = 2$, since $H_3^4 = \{4, 5, 6\}$ and $H_3^5 = \{2, 5, 6\}$: $P_1 = [7, 2]$, $P_2 = [8, 3]$, $P_3 = [9, 4]$, $P_4 = [10, 4, 7, 2, 8, 3, 9, 10, 4]$, $P_5 = [11, 2, 7, 3, 8, 4, 9, 11, 2]$, $P_6 = [12, 2, 7, 3, 8, 4, 9, 12]$. Step 2 of Stage 2 will find that paths P_4, P_5, and P_6 have a common internal node (the last node of P_6 is an internal node of P_4 and P_5), and these nodes have the same H_2. We change i_4 and i_5 (since v_4 and v_5 have odd distance from u) such that adding these to the corresponding H_1s will result in different sets which are also different from H_1^6. One possibility is to use $i_4 = 5$ and $i_5 = 6$, since $\{2, 3, 5\}$, $\{3, 4, 6\}$, and $\{2, 3, 4\}$ are all different (we explain how to find these elements in Theorem 1). Then paths P_4 and P_5 are redefined to get $P_4 = [10, 5, 7, 2, 8, 3, 9, 10, 5]$, $P_5 = [11, 6, 7, 3, 8, 4, 9, 11, 6]$, $P_6 = [12, 2, 7, 3, 8, 4, 9, 12]$, making the paths internally node disjoint, and the algorithm terminates.

Now we prove our main result.

Theorem 1. *The Path Algorithm will finish in $O(n^5)$ time, and the resulting paths will be pairwise internally node-disjoint.*

Proof. Since the preliminary step and Stage 1 can be done in $O(n^3)$ time, it is enough to show that each step of Stage 2 will finish in $O(n^5)$ time, and that the resulting paths will be internally node disjoint.

Consider first Step 1 of Stage 2. By Lemma 3, we can easily check whether any two paths falling under this case have a common node apart from u by checking initial subsequences of the sequences representing these paths. One pair of paths can be checked in $O(n)$ time, and since we have $O(n^2)$ pairs of paths, this can be done in $O(n^3)$ time. Now, assume that we find two paths $P_\beta = [p_1, p_2, \ldots, p_\gamma]$ and $Q_\delta = [q_1, q_2, \ldots, q_\eta]$ having a common node which is not the endpoint of both paths. By Lemma 3, there must be an initial subsequence of the same length of P and Q containing the same numbers in a different order. Thus there is an $i \le \min\{\gamma, \eta\}$ such that $\{p_1, p_2, \ldots, p_i\} = \{q_1, q_2, \ldots, q_i\}$. Consider only the elements of these paths in H_2 (i.e., with an odd subscript), and without loss of generality assume that $p_1 < q_1$. Let j be the largest odd integer that is not bigger than i. Then we have $\{p_1, p_3, \ldots, p_j\} = \{q_1, q_3, \ldots, q_j\}$ as well. Since $q_1 \in \{p_1, p_3, \ldots, p_j\}$, there is an odd index k such that $p_k = q_1$, and similarly $p_1 \in \{q_1, q_3, \ldots, q_j\}$ implies that there is an odd index m such that $q_m = p_1$. Then $\{p_1, p_3, \ldots, p_j\} = \{q_1, q_3, \ldots, q_j\}$ implies that $j \ge k$ and $j \ge m$. Since these paths are obtained by cyclically permuting the corresponding H_2, we get that $p_1 < p_3 < \cdots < p_{k-2} < p_k = q_1$, and we must have $q_m = p_1, q_{m+2} = p_3, \ldots, q_j = p_{k-2}$ (in particular, $j - m = k - 3$). Similarly, we get that $q_1 = p_k, q_3 = p_{k+2}, \ldots, q_{m-2} = p_j$. So this implies that H_2^{β} has no elements between $q_j = p_{k-2}$ and p_k, and H_2^{δ} has no elements between $p_j = q_{m-2}$ and q_m (betweenness is meant cyclically, so, if $q_{m-2} \geq q_m$, then this means that H_2^{δ} has no elements bigger than q_{m-2} and no elements less than q_m). See Fig. 2 for an example in HS(48, 24), where $k = 7, m = 11, j = 15$ (various p_i and q_i are indicated below or above the corresponding value).

So, if $H_2^{\beta} \neq H_2^{\delta}$, then either H_2^{β} has an element between $q_{m-2} = p_j$ and p_1, or H_2^{δ} has an element between $p_{k-2} = q_j$ and q_1 (33 in H_2^{β} and 42 in H_2^{δ} in the above example). In either case, switching the chosen bits will make the redefined paths to have no common internal nodes. Note that whenever we do a switch and $H_2^{\beta} \neq H_2^{\delta}$, the number of cyclically missing elements in H_2^{β} and in H_2^{δ} after the corresponding last element and before the corresponding first element of $CR_x(H_2^{\beta})$ and $CR_x(H_2^{\delta})$ will go up (a missing element corresponds to a 0 in the matrix M_2): originally, it is strictly less than $(p_1 - p_j - 1) + (q_1 - q_j - 1)$ (since $H_2^{\beta} \neq H_2^{\delta}$, there is an element either in H_2^{β} after p_j or in H_2^{δ} after q_j); after the switch, it becomes $(p_k - p_{k-2} - 1) + (q_m - q_{m-2} - 1) = (q_1 - q_j - 1) + (p_1 - p_j - 1)$, since H_2^{β} has no elements between p_{k-2} and p_k, and H_2^{δ} has no elements between q_{m-2} and q_m (here again this is meant cyclically, so, if $p_j = q_{m-2} > q_m = p_1$, then we mean that H_2^{δ} has no elements larger than q_{m-2} or smaller than q_m, so $p_1 - p_j - 1$ is replaced by $n + p_1 - p_j - 1$, and $q_m - q_{m-2} - 1$ is replaced by $n + q_m - q_{m-2} - 1$). In the above example, originally there is no missing element in H_2^{β} between $p_{17} = 33$ and $p_1 = 34$, and there is one missing element (43) in H_2^{δ} between $q_{17} = 42$ and $q_1 = 44$. After the switch, three elements are missing from H_2^{β} between 41 and 44, and three elements are missing from H_2^{δ} between 31 and 34.

Thus, if we add up the number of such missing elements for each node falling under Case 1 of Stage 2, this number will increase after each switch, and is bounded from above by the number of zeros in matrix M_2, so, after at most $O(n^2)$ iterations this process must end, and the only remaining conflicts may occur for nodes with the same H_2. Thus this part of the algorithm can be done in $O(n^5)$ time. As another illustration, consider Example 2. Originally there is one missing element: $CR_0(H_2^1) = [5, 7]$ is missing 8, but $CR_1(H_2^2) = [6, 5]$ and $CR_2(H_2^3) = [7, 5, 6]$ have no missing element cyclically before their first and after the last elements. After the first switch, we get two missing elements: $CR_1(H_2^1) = [7, 5]$ is missing 6, and $CR_0(H_2^3) = [5, 6, 7]$ is missing 8. After the second switch, we get three missing elements: $CR_1(H_2^1) = [7, 5]$ is missing 6, $CR_0(H_2^2) = [5, 6]$ is missing 7 and 8, and $CR_1(H_2^3) = [6, 7, 5]$ has no missing element.

Now, consider all nodes $v_{j_1}, v_{j_2}, \ldots, v_{j_k}$ with the same corresponding H_2 and chosen bit 1, and assume that among them $v_{j_1}, \ldots, v_{j_m}$ are of even distance from u (which is $2|H_2|$), and the others ($k - m$ nodes) are of an odd distance from u (which is $2|H_2| - 1$). From the argument above, we can see that between two such nodes the only conflict can occur after all elements of H_2 are encountered. For nodes $v_{j_{m+1}}, \ldots, v_{j_k}$, this means we reach the node, so the common node is not an internal node. So we can only have a conflict between nodes $v_{j_1}, \ldots, v_{j_m}$ or between a node from $v_{j_1}, \ldots, v_{j_m}$ and a node from $v_{j_{m+1}}, \ldots, v_{j_k}$. To get rid of all conflicts we need to permute the corresponding H_1s for nodes $v_{j_1}, \ldots, v_{j_m}$ such that excluding the last elements they will be all different from each other and all H_1s corresponding to $v_{j_{m+1}}, \ldots, v_{j_k}$ (these could be the same). To achieve that, construct a bipartite graph as follows. The left side will have nodes corresponding to nodes $v_{j_1}, \ldots, v_{j_m}$ (m nodes in total), and the right side will correspond to subsets of $\{2, \ldots, n\}$ that can be obtained by removing exactly one element from each of $H_1^{j_1}, \ldots, H_1^{j_m}$ in every possible way (at most $m|H_1^{j_1}| = O(n^2)$ nodes). Connect each node on the left to those subsets on the right that can be obtained by removing one element from the corresponding H_1, but delete those subsets that appear among the H_1s corresponding to nodes $v_{j_{m+1}}, \ldots, v_{j_k}$. Since we chose a bit 1 for each of these nodes, we have $|H_2| \geq k$, so the degree of each node on the left is at least m ($|H_1| = |H_2|$ for these nodes, so there are at least k edges originally, and we exclude at most $k - m$ of them). Since there are only m nodes on the left side, Hall's condition will automatically be satisfied, since every nonempty subset of nodes on the left will have at least m neighbors on the right, so the graph has a matching saturating the left side. The degree of each node on the left is at most n, so the number of edges in this bipartite graph is $O(n^2)$; thus a maximum matching can be found in $O(n^4)$ time. Now, permute H_1 for each node $v_{j_1}, \ldots, v_{j_m}$ such that the excluded element in this matching becomes the last element. Since this may need to be done only once for nodes with the same H_2, this part can be done in $O(n^4)$ time. To illustrate this part of the algorithm, consider Example 3. There is a conflict with the initial paths and the corresponding H_2s are the same, so we construct an auxiliary bipartite graph as follows. The left side has nodes {2, 3, 4} and {2, 3, 5} corresponding respectively to v_2 and v_3 (these nodes have even distance from u), the right side has nodes {2, 3}, {2, 4}, {3, 4}, {2, 5}, {3, 5} (sets obtained from the left side by deleting one element), and we have edges from {2, 3, 4} to {2, 3}, {2, 4}, {3, 4}, and from {2, 3, 5} to {2, 3}, {2, 5}, {3, 5} (to the sets obtained by deleting one element). Finally, we delete node {2, 3} on the right side, since v_1 has $H_1^1 = \{2, 3\}$. This leaves four nodes on the right and four edges overall. A matching saturating the left side can easily be found; the choice of edges {2, 3, 4}–{2, 4} and {2, 3, 5}–{3, 5} leads to the solution in Example 3 (we permute the H_1s such that the deleted number becomes its last element, here 3 for v_2 and 2 for v_5).

Thus Step 1 will eliminate all conflicts between nodes falling under Case 1 of Stage 1 in $O(n^5)$ time.

Next consider the case when node v_{β} falls under Case 1 and node v_{δ} falls under Case 2, and assume that the two corresponding paths $P_{\beta} = [p_1, p_2, \ldots, p_{\gamma}]$ and $Q_{\delta} = [q_1, q_2, \ldots, q_{\eta}]$ have a common internal node. Let this common node correspond to the initial subsequences $[p_1, p_2, \ldots, p_i]$ and $[q_1, q_2, \ldots, q_j]$. Then the definitions of these paths imply that $p_1, p_2, \ldots, p_{\gamma}$ are all different, $q_1 = q_{\eta}$, and $q_1, q_2, \ldots, q_{\eta-1}$ are all different. Thus, if the initial subsequences of these paths lead to the same node, then p_1 must appear among $q_2, \ldots, q_{\eta-1}$ (note that $p_1 \neq q_1$ since the first edges on the paths are chosen to be all different). If q_1 also appears among $p_2, \ldots, p_\gamma$, then switching the two chosen bits for v_β and v_δ would increase the number of 1s chosen, so this is not possible. Hence, no initial subsequence of P_β switches bit q_1, so the only way the two paths end up at the same node is if $j = \eta$, so q_1 is switched twice. Then by Lemma 3 (and the remark immediately after that) we must have $\{p_1, p_2, \ldots, p_i\} = \{q_2, \ldots, q_{\eta-1}\}$, so $i = \eta - 2$. If $i = \gamma$, then the only common nodes on the two paths are their endpoints; otherwise $\mathrm{dist}(u, v_\beta) = \gamma > i = \eta - 2 = \mathrm{dist}(u, v_\delta)$, so in the initial phase we would have chosen bit q_1 for node v_δ rather than v_β, so this is not possible.

The next case is when both v_β and v_δ fall under Case 2. Then $p_1 = p_\gamma$ and $q_1 = q_\eta$, but $p_1, p_2, \ldots, p_{\gamma-1}$ are all different, and $q_1, q_2, \ldots, q_{\eta-1}$ are all different. If p_1 appears among $q_2, \ldots, q_{\eta-1}$, then switching the two chosen bits for v_β and v_δ would increase the number of chosen 1s, so this is not possible. Similarly, q_1 cannot appear among $p_2, \ldots, p_{\gamma-1}$. Thus the only way the two paths end up at the same node is if p_1 is switched twice in the first path, and q_1 is switched twice in the second path; thus $i = \gamma$ and $j = \eta$. But then again the only common nodes on the two paths are their endpoints, so there are no internal common nodes on these paths, giving a contradiction. Thus after Step 1 there will be no conflict between nodes falling under either Case 1 or 2.

Next, consider Step 2 of Stage 2. Clearly, we can check in $O(n^3)$ time if there is a conflict by checking initial subsequences of the paths. If there is a conflict, we can find the corresponding i_ks as follows. Let the nodes $v_{j_1}, v_{j_2}, \ldots, v_{j_k}$ have the same corresponding H_2 and chosen bit 0, and assume that among them $v_{j_1}, \ldots, v_{j_m}$ are of odd distance from u (which is $2|H_2|-1$), and the others ($k-m$ nodes) are of an even distance from u (which is $2|H_2|$). Construct an auxiliary bipartite graph as follows. The left side will have nodes corresponding to nodes $v_{j_1}, \ldots, v_{j_m}$ (m nodes in total), and the right side will correspond to subsets of $\{2, \ldots, n\}$ that can be obtained by adding exactly one element from each of $H_3^{j_1}, \ldots, H_3^{j_m}$ in every possible way to the corresponding H_1 (at most $m|H_3^{j_1}| = O(n^2)$ nodes). Connect each node on the left to those subsets on the right that can be obtained be adding one element from the corresponding H_3, but exclude those subsets that appear among the H_1s corresponding to nodes $v_{i_{m+1}}, \ldots, v_{i_k}$. Since the 1s in H_2 for these nodes were not chosen, there are at least $|H_2|$ other nodes, so $k \leq n - |H_2|$. Since $|H_1| = |H_2| - 1$ for nodes $v_{j_1}, \ldots, v_{j_m}$, we get $|H_1| \leq n - 1 - k$, so the original degree of each of these nodes will be at least $|H_3| = (n-1) - |H_1| \geq (n-1) - (n-1-k) = k$, and, after excluding possibly $k - m$ of these, the degree will be at least m. Since there are m nodes on the left, again Hall's condition will automatically be satisfied. The degree of each node on the left is at most n, so the number of edges in this bipartite graph is $O(n^2)$, thus a matching saturating the left side can be found in $O(n^4)$ time, and the numbers i_k for Case 3 can be chosen accordingly. Since a maximum matching needs to be found at most once for nodes with the same H_2, this part can be done in $O(n^4)$ time.

To illustrate this part of the algorithm, consider Example 4. There is a conflict with the initial paths and the corresponding H_2s are the same, so we construct an auxiliary bipartite graph as follows. The left side has nodes $\{2, 3\}$ and $\{3, 4\}$, corresponding respectively to v_4 and v_5 (these nodes have odd distance from u), the right side has nodes $\{2, 3, 4\}, \{2, 3, 5\}, \{2, 3, 6\}, \{3, 4, 5\}, \{3, 4, 6\}$ (sets obtained from the left side by adding one element), and we have edges from $\{2, 3\}$ to $\{2, 3, 4\}, \{2, 3, 5\}, \{2, 3, 6\}$, and from $\{3, 4\}$ to $\{2, 3, 4\}, \{3, 4, 5\}, \{3, 4, 6\}$ (to the sets obtained by adding one element). Finally, we delete node $\{2, 3, 4\}$ on the right side, since v_6 has $H_1^1 = \{2, 3, 4\}$. This leaves four nodes on the right and four edges overall. A matching saturating the left side can easily be found; the choice of edges $\{2, 3\}$–$\{2, 3, 5\}$ and $\{3, 4\}$–$\{3, 4, 6\}$ leads to the solution in Example 4 (the added element is used as i_k; here, $i_4 = 5$ and $i_5 = 6$).

Now, consider the remaining cases, when at least one of v_β and v_δ falls under in Case 3 in Stage 1. Without loss of generality, we may assume that v_δ falls under Case 3, so $q_1, q_2, \ldots, q_{\eta-2}$ are all different, $q_1 = q_{\eta-1}$, and $q_2 = q_\eta$. Again, assume that the corresponding paths $P_\beta = [p_1, p_2, \ldots, p_\gamma]$ and $Q_\delta = [q_1, q_2, \ldots, q_\eta]$ have a common internal node corresponding to the initial subsequences $[p_1, p_2, \ldots, p_i]$ and $[q_1, q_2, \ldots, q_j]$. We will consider cases depending on which cases v_β falls under in Stage 1. First, assume that v_β falls under Case 1, so $p_1, p_2, \ldots, p_\gamma$ are all different. Thus p_1 must appear among $q_2, \ldots, q_{\eta-2}$ for the paths to have a common node. If q_1 also appear among $p_2, \ldots, p_\gamma$, then switching the two chosen bits for v_β and v_δ would increase the number of 1s chosen, so this is not possible. Hence no initial subsequence of path P_β switches bit q_1, so the only way to have a common node is if q_1 is switched twice, so $j = \eta - 1$ or $j = \eta$. By Lemma 3 (and the remark immediately after that), we must have $i = j - 2$ or $i = j - 4$; thus in both cases $\mathrm{dist}(u, v_\beta) = \gamma \geq i \geq j - 4 = \mathrm{dist}(u, v_\delta)$. If $\mathrm{dist}(u, v_\beta) > \mathrm{dist}(u, v_\delta)$, then in the initial phase we would have chosen bit q_1 for node v_δ rather than v_β, so $\mathrm{dist}(u, v_\beta) = \mathrm{dist}(u, v_\delta)$, which implies that $i = \gamma$ and $j = \eta$, so there is no common internal node on the two paths, which is a contradiction.

If v_β falls under Case 2, then $p_1 = p_\gamma$, and $p_1, p_2, \ldots, p_{\gamma-1}$ are all different. If either p_1 occurs among $q_3, \ldots, q_{\eta-2}$, or q_1 occurs among $p_2, \ldots, p_{\gamma-1}$, then switching the two chosen bits increases the number of chosen 1s, so this is not possible. So for the paths to have a common node, both p_1 and q_1 must be switched twice, so $i = \gamma$, and $j = \eta - 1$ or $j = \eta$. If $j = \eta$, then the paths only have their endpoints in common, so $j = \eta - 1$. This implies that $H_2^\beta = H_2^\delta$ and $H_1^\beta = H_1^\delta \cup \{q_2\}$, which is not possible, since q_2 is chosen in Step 2 of Stage 2 such that $H_1^\delta \cup \{q_2\}$ is different from the set H_1 of every other node with the same H_2.

Finally, if v_β falls under Case 3, then $p_1, p_2, \ldots, p_{\gamma-2}$ are all different, $p_1 = p_{\gamma-1}$, and $p_2 = p_\gamma$. Again, neither p_1 nor q_1 can occur in the other path, otherwise switching the chosen bits increases the number of chosen 1s. Thus the only way for the paths to have a common node is if both bits are switched twice, so $i = \gamma - 1$ or $i = \gamma$, and $j = \eta - 1$ or $j = \eta$. This leaves four possibilities. If $i = \gamma$ and $j = \eta$, then the common node on the paths is their common endpoint. If $i = \gamma$ and $j = \eta - 1$, then one of the paths leads to a node of even distance from u, and the other of an odd distance, which is impossible. The case

$i = \gamma - 1$ and $j = \eta$ is impossible for the same reason. This leaves $i = \gamma - 1$ and $j = \eta - 1$, which implies that $H_2^\beta = H_2^\delta$ and $H_1^\beta \cup \{p_2\} = H_1^\delta \cup \{q_2\}$, which is not possible, since p_2 and q_2 are chosen in Step 2 of Stage 2 such that $H_1^\beta \cup \{p_2\}$ and $H_1^\delta \cup \{q_2\}$ are different. This is a contradiction. Since each step can be done in $O(n^5)$ time, the proof is finished. □

Note that, for node v_i falling under Case 1 of Stage 1, the path we obtain is a shortest path; for nodes falling under Case 2, the path has length $\text{dist}(u, v_i) + 2$, while, for nodes falling under Case 3, the path has length $\text{dist}(u, v_i) + 4$. So the maximum length of the path obtained by the algorithm is $2n + 1$, when $\widetilde{r_{k,j}} = 0$ and $\text{dist}(u, v_i) = 2n - 3$.

Remark. Every path obtained will be a shortest path if and only if we can choose bit 1 from M_2 for every node in the beginning, that is, there is a maximum matching in the corresponding auxiliary graph.

4. Conclusion

In this paper, we have provided an algorithm to construct one-to-many internally node-disjoint paths in $\text{HS}(2n, n)$ in time polynomial in n. We note that the corresponding disjoint one-to-many shortest paths routing problem for the hypercube is solved in [8]. Interestingly, its algorithm also involves finding a perfect matching. On the other hand, this is not surprising, since $\text{HS}(2n, n)$ is a subgraph of the hypercube. We would like to point out that the algorithm given here is much more involved than the one given in [8]. So here is another example of it being more difficult to prove properties for $\text{HS}(2n, n)$ than for the hypercube (just like the Hamiltonian problem). In [8], a necessary and sufficient condition is given for each of the n paths to be shortest. Here, we gave a corresponding condition in the Remark at the end of the previous section.

Acknowledgments

We would like to thank the anonymous referees for their helpful comments, which greatly improved this paper.

References

[1] S.B. Akers, D. Harel, B. Krishnamurthy, The star graph: an attractive alternative to the n-cube, in: Proceedings of the 1987 International Conference on Parallel Processing, 1987, pp. 393–400.
[2] E. Cheng, S. Gao, K. Qiu, Z. Shen, On disjoint shortest paths routing on the hypercube, in: D.-Z. Du, X. Hu, P.M. Pardalos (Eds.), Proceedings of the 3rd Annual International Conference on Combinatorial Optimization and Applications, in: Lecture Notes in Computer Science, vol. 5573, Springer-Verlag, 2009, pp. 375–383.
[3] E. Cheng, P. Hu, R. Jia, L. Lipták, Matching preclusion and conditional matching preclusion for bipartite interconnection networks II: Cayley graphs generated by transposition trees and hyper-stars, Networks (2011), in press (http://dx.doi.org/10.1002/net.20441).
[4] E. Cheng, L. Lipták, Structural properties of hyper-stars, Ars Combinatoria 80 (2006) 65–73.
[5] E. Cheng, M. Shah, A strong structural theorem for hyper-stars, Congressus Numerantium 179 (2006) 181–191.
[6] D.R. Duh, G.H. Chen, Topological properties of WK-recursive networks, Journal of Parallel and Distributed Computing 23 (3) (1994) 468–474.
[7] J.S. Fu, Longest fault-free paths in hypercubes with vertex faults, Information Sciences 176 (7) (2006) 759–771.
[8] S. Gao, B. Novick, K. Qiu, From Hall's matching theorem to optimal routing on hypercubes, Journal of Combinatorial Theory (Series B) 74 (2) (1998) 291–301.
[9] Q.P. Gu, S.T. Peng, An efficient algorithm for the k-pairwise disjoint paths problem in hypercubes, Journal of Parallel and Distributed Computing 60 (6) (2000) 764–774.
[10] D.F. Hsu, On container width and length in graphs, groups, and networks, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E77-A (4) (1994) 668–680.
[11] S.L. Johnson, C.T. Ho, Optimal broadcasting and personalized communication in hypercubes, IEEE Transactions on Computers 38 (9) (1989) 1249–1268.
[12] J.-S. Kim, E. Cheng, L. Lipták, H.-O. Lee, Embedding hypercubes, rings and odd graphs into hyper-stars, International Journal of Computer Mathematics 86 (5) (2009) 771–778.
[13] J.-S. Kim, E. Oh, H.-O. Lee, Y.-N. Heo, Topological and communication aspects of hyper-star graphs, in: Proceedings of the 18th International Symposium on Computer and Information Sciences, in: LNCS, vol. 2869, 2003, pp. 51–58.
[14] C.N. Lai, G.H. Chen, Strong Rabin numbers of folded hypercubes, Theoretical Computer Science 341 (1–3) (2005) 196–215.
[15] H.-O. Lee, J.-S. Kim, E. Oh, H.-S. Lim, Hyper-star graph: a new interconnection network improving the network cost of hypercube, in: Proceedings of EurAsia ICT: Information and Communication Technology, in: LNCS, vol. 2510, 2002, pp. 858–865.
[16] T. Leighton, Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes, Morgan Kaufman, San Mateo, CA, 1992.
[17] T.-C. Lin, D.-R. Duh, Constructing vertex-disjoint paths in (n, k)-star graphs, Information Sciences 178 (2008) 788–801.
[18] M.O. Rabin, Efficient dispersal of information for security, load balancing, and fault tolerance, Journal of the ACM 36 (2) (1989) 335–348.
[19] Y. Saad, M.H. Schultz, Topological properties of hypercubes, IEEE Transactions on Computers 37 (7) (1988) 867–872.
[20] R.-Y. Wu, G.-H. Chen, Y.-L. Kuo, G.J. Chang, Node-disjoint paths in hierarchical hypercube networks, Information Sciences 177 (2007) 4200–4207.