Available online at www.sciencedirect.com
SCIENCE DIRECT®
Parallel Computing 30 (2004) 973–998
PARALLEL COMPUTING
www.elsevier.com/locate/parco

n-Cube network: node disjoint shortest paths for maximal distance pairs of vertices ☆

Teofilo F. Gonzalez *, David Serena

Department of Computer Science, University of California at Santa Barbara, Santa Barbara, CA 95064, United States

Received 21 December 2002; revised 20 July 2004; accepted 22 July 2004

Abstract

In parallel and distributed systems many communications take place concurrently, so the routing algorithm as well as the underlying interconnection network play a vital role in delivering all the messages efficiently. Fault tolerance and performance are often obtained by delivering the messages through node disjoint shortest paths. In this paper we present two efficient algorithms to construct, under certain conditions, pairwise node disjoint shortest paths for pairs of vertices in an n-cube in the presence of faulty nodes. The first algorithm has $O(m^2)$ time complexity, where m is the number of input bits, and the second one takes $O(m^3)$, but it solves more general problem instances. We also present an efficient algorithm for the extreme version of the edge disjoint shortest paths problem when n is odd.

Keywords: Fault-tolerance; Hypercube; n-cube; Pairwise node disjoint shortest paths; Routing

1. Introduction

The n-cube is a fundamental structure for parallel computing. Several systems with this communication architecture have been built. The SGI Origin computer

☆ Preliminary versions, without the proofs, of this work appear in [2,5].

* Corresponding author.
E-mail addresses: teo@cs.ucsb.edu (T.F. Gonzalez), dserena@cs.ucsb.edu (D. Serena).

0167-8191/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.parco.2004.07.006

system is a recent computing platform whose interconnection network is a variation of the n-cube. There are many algorithms for several different routing problems that arise while executing code on an n-cube connected machine. In this paper we present efficient algorithms for routing problems which are common to many applications. Our problem has applications when network traffic endpoints are defined by empirically observed flows; or are specifically initiated by the individual nodes in the network to designated destinations.

The *p-pairwise node disjoint shortest paths problem* for the n-cube is given p pairs of nodes and q blocking nodes denoted by

$$X = \{X_1, X_2, \ldots, X_p, X_{p+1}, X_{p+2}, \ldots, X_{p+q}\},$$

where $X_i = (s_i, t_i)$, for $1 \leqslant i \leqslant p$, and $X_i = (a_i)$ for $p + 1 \leqslant i \leqslant p + q$, find node disjoint shortest paths in the n-cube for all the pairs X_i, i.e. the paths do not include blocking nodes and no two such paths have a node in common. Each pair $X_i = (s_i, t_i)$ consists of two *endpoints* which are called the *source* and *target,* respectively. The nodes a_i (or blocking nodes or faulty processors) may also be included as part of the input. Every node in the n-cube is represented by an n-bit string and there is an edge between two nodes if their bit representation disagrees in exactly one bit. The distance between the source and target nodes of pair X_i (or *pair distance*) in the n-cube is denoted by $d(X_i) = d(s_i, t_i)$ and it is the number of bits that differ in the bit representation of s_i and t_i. The distance $d(a,b)$ is frequently referred to as the Hamming distance between nodes a and b in the n-cube. By a *shortest path* for the pair X_i we mean any path from s_i to t_i with length equal to $d(X_i)$, i.e. the path must be a shortest path in the graph between the two nodes independent from any other paths, blocking nodes or endpoints of the other pairs. The *edge disjoint shortest paths problem* is given X (without faulty nodes or $q = 0$) in the n-cube, find shortest paths connecting each s_i to t_i such that no two paths have an edge in common. The edge disjoint shortest paths problem is said to be a *partial half permutation routing request* because every vertex in the n-cube is occupied by a source or target, but not both. However, the node disjoint shortest paths problem defined above is said to be a *partial half permutation routing request with singletons*.

Both the decision problem and the search problem are important for analysis. In the (undirected) n-cube there are "yes" and "no" instances of the k-pairwise node disjoint shortest paths problem. The algorithms presented herein are search algorithms in the sense that they construct for yes instances a set of node disjoint shortest paths.

In the context of undirected graphs the order of the source to target in the routing request is irrelevant. Therefore, we consider the undirected pairs $X_i = \{s_i, t_i\}$, $1 \leqslant i \leqslant p$, in lieu of the directed pairs $X_i = (s_i, t_i)$, $1 \leqslant i \leqslant p$. Directed pairs are relevant for directed n-cubes and graphs. In the 2-cube $X = \{\{00, 11\}, \{01\}, \{10\}\}$ has no solution because every (shortest) path between 00 and 11 must go through 01 or 10. On the other hand, $Y = \{\{00, 11\}, \{01\}\}$ does have a solution with pair $\{00, 11\}$ yielding route $00 \leftrightarrow 10 \leftrightarrow 11$. For problem instance Y given above the only possible routing generated by a search algorithm would be $00 \leftrightarrow 10 \leftrightarrow 11$. Note that

while problem instance $Z = \{\{000, 011\}, \{001\}, \{010\}\}$ has no shortest path solution, it does have a routing with arbitrary length paths: $000 \leftrightarrow 100 \leftrightarrow 101 \leftrightarrow 111 \leftrightarrow 011$. To clarify the problem the possible shortest path routings are $000 \leftrightarrow 010 \leftrightarrow 011$ and $000 \leftrightarrow 001 \leftrightarrow 011$. Though these paths are shortest paths by our definition, neither of these paths may be traversed due to the blocking nodes in the input: $\{001\}$ and $\{010\}$. For edge disjoint shortest paths both of those paths are possible, because we do not have blocking nodes. However the problem instance $Z = \{\{00, 11\}, \{10, 01\}\}$ does not have edge disjoint shortest paths.

In this paper we present efficient algorithms for versions of the k-pairwise node disjoint shortest path problem, as well as for the k-pairwise edge disjoint shortest path problem, in the n-cube. Before we outline our results in more detail we discuss previous work related to our problems.

Many of the message routing problems mentioned above are known to be computationally difficult for general graphs when one allows arbitrary length paths, rather than just shortest ones. Karp [9] analyzes the k-pairwise disjoint paths problem in general graphs and establishes NP-completeness. Shiloach [15] presented a polynomial time algorithm to construct node disjoint paths in a graph for two pairs of vertices and Watkin [16] showed that $(2k - 1)$-connectedness is a necessary condition for a graph to admit disjoint paths for a set of k pairs of vertices.

The related problem where one seeks to find p-pairwise node disjoint paths (*arbitrary distance pairs*) from vertices in set $\{s_1, s_2, \ldots, s_p\}$ to vertices in set $\{t_1, t_2, \ldots, t_p\}$ is called the set-to-set node disjoint paths problem. In this problem one needs to find p node disjoint paths from one set to the other such that the paths are from s_i to $t_{\phi(i)}$ where ϕ:$\{1, 2, \ldots, p\} \rightarrow \{1, 2, \ldots, p\}$ and ϕ is any 1–1 function. Whereas in the k-pairwise problem ϕ is the identity function, namely $\phi(i) = i$. The undirected vertex version of Menger's theorem is applicable to the former problem, but not the latter [13]. It is not applicable to the set-to-set node disjoint *shortest* paths problem as Menger's Theorem may imply *arbitrary* length paths. Note that for the arbitrary length k-pairwise node disjoint paths problem a separating set of nodes of degree k need not imply that k pairwise node disjoint paths exist. The counter example in Fig. 1(b) shows that although the separating set for vertices has cardinality two, there do not exist two pairwise node disjoint paths for the routing request $X = \{\{s_1, t_1\}, \{s_2, t_2\}\}$.

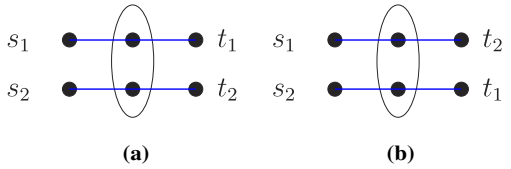

Fig. 1. Even though there is a 2 node separating set (circled) the routing request $X = \{\{s_1, t_1\}, \{s_2, t_2\}\}$ shown in (a) has a solution to the node disjoint paths problem while the graph in (b) does not.

Madhavapeddy and Sudborough [11] developed an $O(n^3 \log n)$ time algorithm to find disjoint paths for k pairs in an n-cube when $2 < k \leqslant \lceil n/2 \rceil$ and $n \geqslant 4$. Note that this algorithm finds arbitrary length paths, but each path is of length at most $2n$. Subsequently, Gu and Peng [6] presented an algorithm that takes $O(kn \log n)$ time to find node disjoint paths for k pairs; even if there are $n - 2k + 1$ faulty clusters of diameter 1, and the value of k is at most $\lceil n/2 \rceil$.

The main difference between the work mentioned above and ours is that we are interested in finding either node or edge disjoint *shortest* paths rather than just node or edge disjoint paths (i.e. for each pair X_i the length of the path must be $d(X_i)$). Rabin [14] proposed an algorithmic strategy which addresses fault-tolerance, security and load balancing. While the paper is an interesting starting point, it is clear that in general, three arbitrary parameters of optimization may sometimes be contradictory or at least mutually interdependent.

Gu and Peng [7] address the problem of constructing set-to-set node disjoint paths. The main difference between their problem and ours is that they just need to find a path from each vertex in one set to a (different) vertex in the other set; whereas, in our problem one constructs *shortest* paths for a *given set of pairs* of vertices. Gu and Peng's paper [7] also presents algorithms for the node-to-set node disjoint paths problem. Gao et al. [1] presented an algorithm for finding node-to-set node disjoint *shortest* paths. They exploit the fact that local n-cube constraints, the existence of a first step in a path, dominate in determining the existence of node-to-set node disjoint shortest paths. Namely, the existence of a system of distinct representatives (SDR) is a necessary and sufficient condition for the existence of node disjoint shortest paths. We have observed to some extent, similar characteristics in restricted versions of our problem. Greedy algorithms typically exploit non-path dependent characteristics of a problem. The path search algorithm in [1] can be properly characterized as just such an approach since once the SDR nodes are found one may easily construct the paths. Node-to-set approaches in the n-cube are highly pragmatic in the sense that they have applications in the context of fault-tolerant distributed networks. Latifi et al. [10] also address this issue for the n-cube. The node-to-set problem reduces to the set-to-set problem after finding the first link for each path. Finding k disjoint paths between two nodes in the hypercube has also been studied [6].

Madhavapeddy and Sudborough [12] show that the k-pairwise edge disjoint paths problem in the hypercube is NP-complete. Unfortunately they did not include the proof that the problem is in NP and we were unable to independently validate that proposition. Their problem allows for vertices in the n-cube to be the source and destination for many pairs. Gonzalez and Serena [4,3] recently showed that the k-pairwise edge disjoint *shortest* path problem is NP-complete even for the partial half permutation routing request and $d(X_i) \leqslant 3$. This result has been extended to arbitrary length paths and $d(X_i) \leqslant 3$ [3]. Madhavapeddy and Sudborough [12] conjectured that the k-pairwise node disjoint paths problem is NP-complete. Recently, this problem has been shown to be NP-complete by Gonzalez and Serena [4,3]. They showed that the k-pairwise node disjoint *shortest* path problem is NP-complete even for the partial half permutation routing request and $d(X_i) \leqslant 3$. Since the general

problems are computationally intractable, we present in this paper efficient algorithms for restricted versions of these problems. Gonzalez and Serena [3,4] have also developed polynomial time algorithms for the node (and also the edge) disjoint shortest paths problem when $d(X_i) \leqslant 2$.

Sections 2 and 3 present polynomial time algorithms for the *extreme* version of our problems. The extreme version of the p-pairwise node disjoint shortest paths problem for the n-cube requires that $d(X_i) = n$ for every pair X_i. The extreme version of the p-pairwise edge disjoint shortest paths problem is defined similarly. Remember that the path for each pair X_i must have length equal to $d(X_i)$. For this version of the problems the endpoints of every pair X_i are complements ($\oplus(2^n - 1)$) of each other in the n-cube, i.e., $\bar{s}_i = s_i \oplus (2^n - 1) = t_i$, where "$\oplus$" is the bitwise "exclusive or" operation.

In Sections 2 and 3 we present algorithms for the extreme version of the p-pairwise node disjoint shortest path problem. The algorithm in Section 2 takes $O((p^2 + pq)n^2)$ time, and solves problem instances such that $p \leqslant \lceil n/2 \rceil$ and $2p + q \leqslant n + 1$. The algorithm in Section 3 is for the case when $p + q \leqslant n - 1$, and therefore applies to a larger set of problem instances. This algorithm has time complexity $O((p^2 + p\ q)n^3)$. In Section 4 we present an efficient algorithm for the extreme version of the edge disjoint shortest paths problem. The time complexity for the algorithm is $O(k\ n)$, and works for all $1 \leqslant p \leqslant 2^{n-1}$ and n odd. For the case when n is even we show paths do not always exist for some values of k.

1.1. Notation and definitions

Our previous definitions imply that all the endpoints and blocking nodes must be different, i.e., the cardinality of the set denoted by $e(X)$ is $2p + q$, where

$$e(X) = \{\text{all endpoints and blocking nodes in } X\} = \bigcup_{i=1}^{p+q} X_i.$$

Let us now define some basic terminology for the n-cube or equivalently the n dimensional hypercube. The n-cube is an undirected graph $G = (V, E)$ with vertex set $V = \{0, 1, \ldots, 2^n - 1\}$ represented by a string of bits in the range [1]

$$\underbrace{00\cdots0}_{n_2} = 0 \leqslant v \leqslant 2^n - 1 = \underbrace{11\cdots1}_{n_2}.$$

The undirected edges E are given by the set $\{\{a, b\} | d(a, b) = 1\}$.

Since shortest paths in the n-cube are analyzed in this paper, it is instructive to define a shortest path predicate. $\Gamma(s, t, u)$ is true if and only if there exists a shortest path in the n-cube from s to t inclusive of u. Formally,

$$\Gamma(s, t, u) = (s \oplus t = s \oplus u \text{ bitwise or } u \oplus t),$$

where "$\oplus$" is the bitwise "exclusive or" operation.

[1] Hereafter binary numbers will appear without the subscript "2".

This predicate may be defined equivalently as

$$\Gamma(s,t,u) = (d(s,u) + d(u,t) == d(s,t)).$$

The subcube defined by the source and destination nodes s and t is defined by

$$K(s,t) = \{u \mid \Gamma(s,t,u) \text{ is true}\}.$$

Note that when $s = t$ the subcube is $s = t = u$. Vertices in $K(s,t)$ are all the nodes which reside on a shortest path from s to t, inclusive of the endpoints.

Qualitatively the 1-bits in the "bitwise exclusive or" ($s \oplus t$) operation can be viewed as bits which change as we move from one node to the next along a shortest path from s to t. Note that due to the topology of the n-cube, at each step only one bit changes, as in the "Gray Code".

An alternate representation can be found in the paper [1] where ordered sets are used to denote a transition from a node to the next in a shortest path. For example the path

$$000 \rightarrow 010 \rightarrow 011 \rightarrow 111$$

is denoted by the ordered set O = (1, 0, 2). There are many other equivalent notations in the literature. We utilize this notation as an output format for the algorithm in Section 4.

2. The simple approach: $p \leqslant \lceil n/2 \rceil$ and $2p + q \leqslant n + 1$

In this section we prove that when $n > 2$, $p \leqslant \lceil n/2 \rceil$, and $2p + q \leqslant n + 1$ node disjoint shortest paths exist for the extreme version of our problem. As shown in Section 2.1, our constructive proofs can be easily implemented to take $O((p^2 + pq)n^2)$ time.

Our algorithmic strategy which we call the *1-bit approach* is defined as follows. First we find an integer k which represents the position of one of the bits in the binary representation of the vertices. The bit satisfies the property that for each endpoint $x \in X_i$ in every pair X_i, $1 \leqslant i \leqslant p$, and its neighbor $g(x) = x \oplus 2^k$ (x and $g(x)$ differ only on bit k) in the n-cube are such that all the $g(x)$ nodes are distinct and every $g(x) \notin e(X)$. In Lemma 2 we show that a bit k satisfying this condition always exists when $p + q < n$. The selection of this bit k is very important because we use it to reduce our original problem of finding node disjoint paths in an n-cube for p pairs of vertices with q blocking nodes to two independent problems. One subproblem, which is rather simple, is in the subcube where bit k is zero and the other one in the subcube where bit k is one. The first subproblem consists of finding a path for one pair in the presence of a number of blocking nodes. One isolates this subproblem by making a transition on an endpoint of one pair using bit k. Lemma 1, whose constructive proof is rather simple, establishes that a solution exists for this subproblem. The other subproblem has $p - 1$ pairs of vertices with another set of blocking nodes. This subproblem is solved recursively.

Let us illustrate our 1-bit approach with the following example. The value of n is 3, $X_1 = \{000, 111\}$, and $X_2 = \{100, 011\}$. Transitioning on the bit $k = 2$ is not allowed

because the neighbor of 000 along bit 2 is $100 \in e(X)$, but bit 0 and bit 1 are possible choices for k. Using bit $k = 1$ our approach is to select from X_1 the endpoint $e_1 = 000$ and from X_2 the endpoint $e_2 = 011$. Their corresponding neighbors are $g(e_1) = 010$ and $g(e_2) = 001$. Now the problem is to find a shortest path from 010 to 111 and one from 001 to 100. Since both paths have to go through nodes in which bit one is never changed (the bit is always one for the first subproblem and zero for the second one), it follows that the resulting problems are independent of each other. Therefore, we may refer to the two resulting problems as finishing up the path for X_1 in the subcube $K(010, 111)$ with blocking node 011 and finishing up the path for X_2 in the subcube $K(001, 100)$ with blocking node 000. By deleting bit k the resulting two independent problems reduce to finding a path in $K(00, 11)$ with blocking node 01 and finding a path in $K(01, 10)$ with blocking node 00. Once we find these paths we add the bit just deleted as well as the previous transition introduced on bit k to produce the node disjoint shortest paths in the 3-cube.

Lemma 1 shows that a node disjoint shortest path exists for a pair in the presence of at most $n - 1$ blocking nodes. The proof of the lemma provides us with a fast algorithm for finding a shortest path for the pair.

Lemma 1. *Given X consisting of one pair $X_1 = \{s_1, t_1\}$ with $d(X_1) = n$ and $q \leqslant n - 1$ blocking nodes in an n-cube for all $n \geqslant 1$, a node disjoint shortest path exists for X_1 that does not include any of the q blocking nodes.*

Proof. We prove this lemma by induction for all $n \geqslant 1$. The base case, $n = 1$ trivially holds because $s_1 = 0$, $t_1 = 1$ and $q = 0$, so the path $0 \leftrightarrow 1$ is valid since there are no blocking nodes.

Assume the lemma holds for all values less than or equal to $n - 1$ and let us now prove that it holds for all $n > 1$. Since the interchange of the 0s and 1s along any subset of dimensions does not change the problem, we may assume without loss of generality that $X_1 = \{s_1, t_1\}$, $s_1 = 0$ and $t_1 = 2^n - 1$. Let α_ℓ be the number of blocking nodes $X_j = \{x_j\}$ such that $d(0, x_j) = \ell$. Clearly $\alpha_\ell < n$ for all ℓ. Let ℓ' be the minimum value of ℓ for which $\alpha_\ell \neq 0$. There are two cases: If $\ell' = 1$, then since $\alpha_{\ell'} < n$ and there are n possible neighbors of $s_1 = \{0\}$ in the subcube $K(0, 2^n - 1)$ one may always choose one of the neighbors of s_1 to be the first node in the path from X_1 that we construct. Let this vacant node be 2^k. Since $\alpha_{\ell'}$ is nonzero the subcube $K(2^k, 2^n - 1)$ contains at most $q - \alpha_{\ell'} \leqslant n - 2$ nodes and the resulting problem reduces to finding a path for the pair of nodes $\{2^k, 2^n - 1\}$ in the subcube $K(2^k, 2^n - 1)$ with at most $n - 2$ blocking nodes. The proof for this case follows by the induction hypothesis after eliminating bit position k.

On the other hand, when $\ell' > 1$, one may move from node 0 to a node in level 1 unimpeded. By the topology of the n-cube there are n possible choices: nodes $2^0, 2^1, \ldots, 2^{n-1}$. By definition all the blocking nodes are distinct and are different from $s_1 = 0$ or $t_1 = 2^n - 1$, so it follows that each blocking node has at least one bit with the value zero and another one with the value of one in its binary representation. Let k be the position of a 0-bit of x, where $X_2 = \{x\}$. Since $\ell' > 1$ one may choose node 2^k to be the first node in the path that we construct for X_1.

Since x is not in the subcube defined by $K(2^k, 2^n - 1)$, the resulting problem reduces to finding a shortest path for a pair of nodes $\{2^k, 2^n - 1\}$ in the subcube $K(2^k, 2^n - 1)$ with at most $n - 2$ blocking nodes. The proof follows by the induction hypothesis after eliminating bit k. □

The constructive proof of Lemma 1 can be used to develop an $O(n^2)$ time algorithm to generate the path for the single pair in the presence of at most $n - 1$ blocking nodes. As we said before, we assume without loss of generality that $X_1 = (000\cdots0, 111\cdots1)$. The algorithm builds a vector $(v[1,n])$ with the number of blocking nodes at a distance i, for $1 \leqslant i < n$, from $000\cdots0$ and for each value of i in $[1, n-1]$ it constructs a list of all the blocking nodes at a distance i from $000\cdots0$. If the entry $v[1]$ is nonzero, then the path for pair X_1 makes the first transition to a node that is not a blocking node. Since there are at most $n - 1$ blocking nodes, it follows that such a node exists. The resulting problem is now solved recursively. Otherwise ($v[1] = 0$) we select any blocking node and find the position of one of its 0-bits. Let us say it is position j. Then the first node for the path for pair X_1 is the node 2^j. The resulting problem is solved inductively without having to generate the data structure again. The algorithm can be easily shown to take $O(n^2)$ time.

As we said before, our algorithmic approach is to find a bit k on which it is possible to make a transition. But our selection for such a bit is actually more restricted than we need. This added flexibility simplifies the process of selecting the endpoints where we will be making the transition for each pair. We find a bit k, such that if every endpoint of a pair makes a transition on bit k, then no endpoint of a pair or blocking node will be overlapped after making the transition. Note that in our algorithm we only make a transition on one endpoint for each pair. Transition on the same dimension k for both endpoints of a pair, X_i, in an n-cube would not result in a shortest path.

Lemma 2. *Let $X = \{X_1, X_2, \ldots, X_p\}$ be pairs of vertices inside an n-cube with $d(X_i) = n$ for $1 \leqslant i \leqslant p$, and $\{X_{p+1}, X_{p+2}, \ldots, X_{p+q}\}$ be a set of blocking nodes such that $1 \leqslant p \leqslant p + q < n$ and $n > 2$. There exist at least $n - p - q + 1$ bit position(s) k such that the cardinality of the set*

$$e(X) \bigcup \{s_i \oplus 2^k \mid 1 \leqslant i \leqslant p\} \bigcup \{t_i \oplus 2^k \mid 1 \leqslant i \leqslant p\}$$

is $4p + q$. This means that no two such vertices are identical and thus one may use bit k to be the starting transition for the paths for all pairs.

Proof. First we consider the case wherein $q = 0$ and $p = n - 1$. For all $0 \leqslant l < n$ define the set

$$S_l = \{\{i, j\} \mid x \in X_i, \quad y \in X_j \quad \text{and} \quad x \oplus y = 2^l, \quad 1 \leqslant i \leqslant p, \quad 1 \leqslant j \leqslant p\}.$$

Note that since $n > 2$, we know that $i \neq j$. The meaning of these S_l sets is as follows. If $S_l \neq \emptyset$, then the neighbor of at least one endpoint of $\{X_1, X_2, \ldots, X_p\}$ using the l-bit transition is an endpoint in X. Therefore bit l cannot be used by our algorithm to reduce the problem to two independent problems of the previously established form.

To prove the lemma we show that for any problem instance there are at least $n - p + 1$ empty sets S_l when $p \leqslant n - 1$.

For each l, $0 \leqslant l < n$, such that $S_l \neq \emptyset$ select a tuple $a_l \in S_l$. Since $n > 2$ we know that if $\{i,j\} \in S_l$, then $\{i,j\} \notin S_{l'}$ for all $l' \neq l$ simply because if there exists $x \in X_i$, $y \in X_j$ such that $x \oplus y = 2^l$ and $\bar{x} \oplus \bar{y} = 2^l$ (remember that the bit complement of x is $\bar{x}$), then $d(x,\bar{y}) > 1$ and $d(y,\bar{x}) > 1$. This precludes $\{i,j\}$'s membership in any $S_{l'}, l \neq l'$.

Therefore, all the tuples a_l are distinct. Define the graph G' with the vertex set $V' = \{1,2,\ldots,p\}$ and edge set $E' = \{a_i | 0 \leqslant i < n\}$. We claim that the graph G' does not have a cycle. Let us prove this claim. Suppose there is a cycle C with nodes $h_1,h_2,\ldots,h_r$. Clearly $r \leqslant p < n$. Since there are no self or multiple edges between node(s) in G' we know that $r > 2$. Without loss of generality assume the edge $\{h_1,h_2\}$, with label k_1, exists because $s_{h_1} \oplus s_{h_2} = 2^{k_1}$ and thus $\{h_1,h_2\} \in S_{k_1}$. Now, for $i = 2,3,\ldots,r-2$, we claim the edge $\{h_i,h_{i+1}\}$ exists because $s_{h_i} \oplus s_{h_{i+1}} = 2^{k_i}$. The reason is that the endpoints of pair X_{i+1} can be relabeled. Now for the last edge in the cycle $\{h_r,h_1\}$ there are two possibilities. Either it exists because of $s_{h_r} \oplus s_{h_1} = 2^{k_r}$ or $s_{h_r} \oplus t_{h_1} = 2^{k_r}$.

The former case is impossible because in an n-cube every cycle must make at least two transitions using the same bit, which is not possible in C because each edge is from a different set S_l. In the latter case, we know that a path between the endpoints of pair X_{h_i}, in the n-cube (G) has p edges (or transitions). Since we make each transition on a different bit and $d(X_{h_1})$ is equal to n, it must then be that $p = n$. But $p < n$, so there is a contradiction.

Therefore the graph G' is a tree. Since there are p pairs and hence at most p nodes in V', then the maximum number of edges in G' is $p - 1$. Thus, the number of empty sets S_l is at least $n - p + 1$. Therefore the lemma holds for the case when $q = 0$ and $p = n - 1$. By adding pairs to any given problem input the lemma is seen to hold for $q = 0$ and $p < n$. The lemma holds for the criteria $p + q < n$, since when $q \geqslant 1$ just convert the blocking nodes to pairs. If a is a blocking node, then node $a \oplus (2^n - 1)$ is either a blocking node or not part of the given input $e(X)$. In the latter case the condition trivially holds. In the former the number of pairs in the resultant input is one less. Thus the lemma holds. □

We are now ready to establish the main theorem in this section.

Theorem 1. *Given X consisting of p pairs of nodes and q blocking nodes in an n-cube for $n > 1$. Node disjoint shortest paths exist for all i and j such that $1 \leqslant i \leqslant p < j \leqslant p + q$, $d(X_i) = n$, $d(X_j) = 0$, $p \leqslant \lceil n/2 \rceil$ and $2p + q \leqslant n + 1$.*

Proof. The proof is by induction on n. The base case is when $n = 2$ and it is covered by Lemma 1. Assume the theorem holds for all $n - 1$ and let us prove it for n.

The case when $p = 0$ is trivial and when $p = 1$ it falls under the conditions of Lemma 1 since $2p + q \leqslant n + 1$ implies that $q \leqslant n - 1$. So assume that $p \geqslant 2$.

Using Lemma 2 we know there exists a bit k where every endpoint in a pair can make a transition without a conflict. We make a transition for one pair from 1 to 0

along the kth bit and make $p-1$ transitions from 0 to 1 for the remaining pairs. Therefore, the subproblem in which all the kth bit positions are 0 will have one pair (the one making the transition from 1 to 0) plus the blocking nodes with a zero in the kth bit position plus one blocking node for each of the remaining pairs (the endpoints of such pairs that have a zero in the k bit position). Therefore the resulting problem in the $(n-1)$-cube has 1 pair and q' blocking nodes, where $q' \leqslant n+1-2p+p-1 = n-p < n-1$. This problem can be solved by using our constructive proof for Lemma 1. The other subproblem in which all the kth bit positions are 1 will have the $p-1$ pairs that make the transition from 0 to 1, plus the blocking nodes that have a one in the kth bit plus one endpoint for the pair that makes the transition from 1 to 0. This resulting problem is in the $(n-1)$-cube, has $p' = p-1$ pairs and $q' \leqslant n+1-2p+1$ blocking nodes. Therefore, $p'+q' \leqslant p-1+n+1-2p+1 \leqslant n-1$ which holds for $p \geqslant 2$. Hence, the resulting problem falls into the induction hypothesis. The theorem follows by induction. □

2.1. Algorithm for the simple approach

The following algorithm assumes that solutions are calculated for dimension $n = 2$, forming a base case of the recursion. The algorithm uses the proof of Theorem 1 to partition the problem. The program is called initially with Find_Paths(n, X).

```
Find_Paths(n, X = {X_1, X_2, ..., X_p, X_{p+1}, ..., X_{p+q}})
{
assume that n = d(X_1) = d(X_2) = ··· = d(X_p), d(X_{p+1}) = d(X_{p+2}) = ··· = d(X_{p+q}) = 0,
 all X_i are in the same n dimensional subcube, p ⩽ ⌈n/2⌉ and 2p + q ⩽ n + 1
if p = 0 return;
if p = 1 construct path using the proof of Lemma 1, output and return;
if n ⩽ 2 return appropriate base case;
k ← Find_Bit(X); // using the proof of Lemma 2 find an unobstructed bit.
X' ← OneToZeroTransition(k, X);
X'' ← ZeroToOneTransition(k, X);
Output_Paths(k, X);
Output path for X' using the proof of Lemma 1.
Find_Paths(n − 1, X'');
}
```

Output_Paths(k, X) prints out a link for each path for a given transition on k. OneToZeroTransition selects one path per the proof of Theorem 1 for making the transition from 1 to 0 on bit k; and the blocking nodes (as well as the endpoints of the remaining pairs) with a 0 in the kth bit position are selected. ZeroToOneTransition selects $p-1$ paths for making a transition from 0 to 1 on bit k; and the blocking nodes (as well as the endpoints of the remaining pairs) with a 1 in the kth bit position are selected.

Theorem 2. *Given X consisting of p pairs of nodes and q blocking nodes in an n-cube. If for all i and j such that $1 \leqslant i \leqslant p < j \leqslant p+q$, $d(X_i) = n$, $d(X_j) = 0$, $p \leqslant \lceil n/2 \rceil$, $2p + q \leqslant n + 1$, then algorithm Finds_Paths constructs node disjoint shortest paths for X and can be easily implemented to take $O((p^2 + pq)n^2)$ time.*

Proof. The proof of correctness follows directly from the constructive proof of Theorem 1.

All the steps can be implemented to take $O((p+q)n)$ time except for procedure Find_Bit and the procedure that constructs a path using the proof of Lemma 1. The procedure that constructs a path for one pair using Lemma 1 takes $O(n^2)$ as we have seen just after the proof of that lemma. Procedure Find_Bit is implemented by using a binary trie [2] to represent nodes in the n-cube. The approach is to insert in the trie the $2p$ nodes in the pairs, the q singletons and then insert the $2p$ nodes resulting from making a transition on bit k for each of the endpoints in each pair. If there are no conflicts for the $4p + q$ insert operations, then the bit k for making a transition has been found. Otherwise one needs to try another bit. Since there are n bits and each node consists of an n-bit number, the total time required for this operation will take at most $O(n^2(p+q))$ steps. Therefore the overall time complexity for the algorithm is given by

$$T(n,p,q) = \begin{cases} T(n,p-1,q+1) + cn^2(p+q), & p > 1, \\ n^2, & p = 0. \end{cases}$$

The solution of $T(n,p,q)$ is $O(p\ n^2(p+q))$. □

Since the number of input bits is $m = (2p+q)n$, then the overall time complexity of the algorithm is $O(m^2)$.

2.2. Limitations of the 1-bit approach

We use (n,p,q) to represent all problem instances in an n-cube with p paths and q blocking nodes. Note that this collection of problem instances are not all "yes" or all "no" instances of the p-pairwise node disjoint shortest paths problem. There are problem instances of $(4,2,3)$ that do not have node disjoint paths whereas others do. For example $\{\{0000,1111\}, \{0011,1100\}, \{0101\}, \{1001\}, \{1010\}\}$ does not have a solution for a reason similar to the one given for Example 1 given below. But $\{\{0000,1111\}, \{0011,1100\}, \{0001\}, \{0110\}, \{1001\}\}$ does ($0000 \leftrightarrow 0100 \leftrightarrow 0101 \leftrightarrow 0111 \leftrightarrow 1111$, and $0011 \leftrightarrow 0010 \leftrightarrow 1010 \leftrightarrow 1000 \leftrightarrow 1100$).

For the case when $n = 4$ it is impossible to strengthen our results in the previous subsection because for $p = \lceil n/2 \rceil + 1 = 3$ none of the problem instances have a solution, i.e., node disjoint shortest paths do not exist. One such instance is given in Example 1.

[2] A binary trie is a binary tree in which we locate any n-cube vertex by following its bit sequence representation [8].

Example 1. The instance of $(n,p,q) = (4,3,0)$ with

$$X = \{\{0000, 1111\}, \{1100, 0011\}, \{1010, 0101\}\},$$

does not have a solution in the 4-cube. i.e., node disjoint shortest paths do not exist for this problem instance.

The proof that Example 1 does not have node disjoint shortest paths is simple. We define the *level* of a node as the number of 1-bits in its binary representation. The only two nodes at level 2 that are not endpoints of the pairs in the 4-cube are 1001 and 0110. Therefore every path from 0000 to 1111 must go through one of these two nodes. Assume without loss of generality that the path goes through 0110 (the other case is identical after applying "exclusive or" with 1111 to all the endpoints). There are four possibilities for the path associated with pair X_1 as indicated by all the simple paths from 0000 to 1111 in Fig. 2.

Nodes 0000, 0110 and 1111 must be part of any shortest path for pair X_1. We claim that the path for pair X_2 must visit node 1001. Suppose not. Suppose that the path for X_2 does not visit node 1001. Then the path for pair X_2 must either remain above or below the nodes on level 2 (i.e. at levels 0 and 1, or at levels 3 and 4). Without loss of generality assume it remains above. Inspection of all the possibilities demonstrates that the path must include the node $0000 \in X_1$, but this is not allowed. Therefore every path for pair X_2 must use node 1001. Considering now the path for pair X_3. Clearly it must either remain above or below level 2. But as in the case for pair X_2 there is no node disjoint shortest path of this form for pair X_3. Therefore no solution exists for this problem instance.

However note that there is a bit $k = 3$ for making a transition (i.e. $0000 \oplus 2^3 = 1000$) as noted in the proof of Lemma 2. Applying the 1-bit step we end up with the problems $\{X_1 = \{000, 111\}, X_2 = \{100\}, X_3 = \{010\}\}$, and $\{X'_1 = \{100, 011\}, X'_2 = \{010, 101\}, X'_3 = \{000\}\}$. The former problem has a solution, but the latter problem does not simply because there are not enough vertices in a 3-cube for two paths of length 3 and a blocking node.

One may find stronger conditions than the ones in Theorem 1 when the value of n is larger. Example 2 gives a problem instance with $n = 5$ and $p = 4$ for which node disjoint shortest paths exist while the proof technique behind Theorem 1 does not cover this scenario.

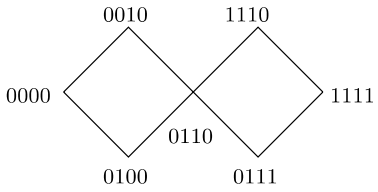

Fig. 2. Remaining possible paths in the 4-cube for routing request $X = \{\{0000, 1111\}, \{1100, 0011\}, \{1010, 0101\}\}$. Note for this request there are no node disjoint shortest paths for all pairs in the request.

Example 2. Consider the instance of $(n,p,q) = (5,4,0)$ with the pairs defined by

$$X = \{\{00000, 11111\}, \{00001, 11110\}, \{00010, 11101\}, \{00100, 11011\}\}.$$

A set of disjoint paths for this problem is given by

$$00000 \leftrightarrow 01000 \leftrightarrow 01001 \leftrightarrow 01011 \leftrightarrow 01111 \leftrightarrow 11111$$
$$00001 \leftrightarrow 00011 \leftrightarrow 00111 \leftrightarrow 00110 \leftrightarrow 01110 \leftrightarrow 11110$$
$$00010 \leftrightarrow 01010 \leftrightarrow 11010 \leftrightarrow 11000 \leftrightarrow 11001 \leftrightarrow 11101$$
$$00100 \leftrightarrow 00101 \leftrightarrow 10101 \leftrightarrow 10111 \leftrightarrow 10011 \leftrightarrow 11011$$

It is interesting to note that our proof technique for Theorem 1 (algorithmic approach) will not work for this problem instance. The reason is that a transition can be made only on bit $k = 3$ or $k = 4$. Since both cases are identical, under interchange of dimensions, we only discuss the case when we make a transition on bit $k = 4$.

When a transition on bit $k = 4$ is made according to our 1-bit step, one pair makes a transition form a 1 to a 0 (pair $\{00000, 11111\}$) and the remaining pairs make a transition from a 0 to a 1 as follows.

$$11111 \rightarrow 01111$$
$$00001 \rightarrow 10001$$
$$00010 \rightarrow 10010$$
$$00100 \rightarrow 10100$$

In the resulting problem the subcube in which bit $k = 4$ is always a one has the pairs

$$X = \{\{0001, 1110\}, \{0010, 1101\}, \{0100, 1011\}\}.$$

"Exclusive or"ing the input with 0001 yields the equivalent problem

$$X = \{\{0000, 1111\}, \{1100, 0011\}, \{1010, 0101\}\}.$$

This is just Example 1 which we know does not have a solution. The reason why our algorithmic approach does not work in this case is that our approach finds a solution with a special structure, but one such solution does not exist for the instance given in Example 2. In general there are problem instances that are not amenable to the 1-bit change approach, so we seek to improve the sufficiency conditions required to insure that node disjoint shortest paths exist for larger values of n. As the number of edges increases exponentially in the n-cube as n increases linearly, one would expect that a much stronger sufficiency condition exists. Section 3 incrementally improves the sufficiency criteria by exploiting an extended version of Lemma 2.

In Example 3 we give a larger problem instance which has a node disjoint shortest path solution, but the algorithm given in this section does not work for this instance. In the next section we give an improved algorithm that works not only for this instance, but for all instances in $(n,p,q) = (7,6,0)$, as well as in more general cases.

Example 3. Instance of $(n,p,q)=(7,6,0)$ with the pairs defined by

$$X = \{\{0000000, 1111111\}, \quad \{0000001, 1111110\},$$
$$\{0000010, 1111101\}, \quad \{0000100, 1111011\},$$
$$\{0001000, 1111011\}, \quad \{0010000, 1101111\}\}.$$

Let us now show that our 1-bit algorithmic strategy does not work for this problem instance. The only bits where transitions are allowed are on bits $k = 5$ and $k = 6$. Suppose we apply first our transformation with bit $k = 6$ and reduce the problem to the instance of $(6,5,1)$ that includes the first five pairs without bit 6. Now the only possible transitions are on bits $k = 4$ and $k = 5$. If we apply our transformation with bit $k = 5$ we reduce the problem to the instance of $(5,4,2)$ that includes the first four pairs without bits 6 and 5. This is the problem instance given in Example 2 with two additional blocking nodes. Since the instance given in Example 2 does not have a solution, neither does this one.

3. The balanced approach: $p + q \leqslant n - 1$

Given n and $X = \{X_1, X_2, \ldots, X_{p+q}\}$ we claim that node disjoint shortest paths exist for the extreme version of the problem when $p + q \leqslant n - 1$ (Theorem 3). An algorithm based on the proof of Theorem 3 is given in Section 3.1.

Consider an instance of the $(8,7,0)$ problem. If we apply the 1-bit approach described in the previous section we end up with instances of $(7,1,6)$ and $(7,6,1)$. Let us suppose now that the instance of $(7,6,1)$ has a bit where a transition is possible. If we apply again the 1-bit approach we end up with instances of $(6,1,7)$ and $(6,5,2)$. Both of these instances do not always have a solution so we cannot prove that this approach is feasible. This is why we have to introduce an additional transformation which we call the *1-bit balanced transformation*. This approach is similar in nature to the 1-bit step given in the previous section. The main difference is that the resulting problems are balanced with respect to p. However this transformation alone is insufficient to make the resulting problems fall into the inductive hypothesis. For example if we start with an instance of $(8,7,0)$ we may end up with two instances of $(7,4,3)$ in the partitioned subproblem. For these instances one cannot guarantee the existence of a bit k as described in Lemma 2. We introduce an additional transformation which we call the *q-dependent transformation* to reduce our problems to ones that fall within the induction hypothesis.

The 1-bit balanced transformation begins by selecting a bit k which we know exists by Lemma 2 since $p < n$. The reduction makes a transition on $\lceil p/2 \rceil$ of the p pairs from 1 to 0 along the kth bit and makes a transition on $\lfloor p/2 \rfloor$ from 0 to 1 on the remaining pairs. The subproblem in which all the kth bit positions are 1 will have the $\lfloor p/2 \rfloor$ pairs that make the transition from 0 to 1, plus one endpoint for each of the $\lceil p/2 \rceil$ pairs that make the transition from 1 to 0. Therefore the resulting problem is an instance of $(n-1, \lfloor p/2 \rfloor, \lceil p/2 \rceil)$. The other subproblem in which all the kth bit positions are 0 will have the $\lceil p/2 \rceil$ pairs that make the transition from

1 to 0, plus one endpoint for each of the $\lfloor p/2 \rfloor$ pairs that make the transition from 0 to 1. This resulting problem is an instance of $(n-1, \lceil p/2 \rceil, \lfloor p/2 \rfloor)$. The resulting subproblems do not fall into the induction hypothesis and do not satisfy the conditions of Lemma 2.

The above situation arises when considering the following instance of $(8,7,0)$:

$$\begin{aligned} X = \{&\{00001111, 11110000\},\\ &\{01000111, 10111000\},\\ &\{01100011, 10011100\},\\ &\{01110001, 10001110\},\\ &\{10110000, 01001111\},\\ &\{10011000, 01100111\},\\ &\{10001100, 01110011\}\}. \end{aligned}$$

Applying the 1-bit balanced transformation we obtain instances of $(7,4,3)$ and $(7,3,4)$. The former instance is obtained by making the transition from 1 to 0 along bit $k = 7$ in the first 4 pairs and the transition from 0 to 1 in the remaining pairs. The resulting problem instance of $(7,4,3)$ is $X = \{\{0001111, 1110000\}, \{1000111, 0111000\}, \{1100011, 0011100\}, \{1110001, 0001110\}, \{1001111\}, \{1100111\}, \{1110011\}\}$. This instance does not have a conflict free bit k as the one which is identified by Lemma 2.

The example above can be easily extended to higher dimensions. To solve our problem we establish in Lemma 3 that a bit k with more restrictive properties always exists and it can be used in an additional transformation which we call the *q-dependent transformation*. The main difference between Lemma 2 and 3 is that in Lemma 3, $p + q$ could be equal to n and in that case a bit k exists, but such a bit may cause a conflict when making one transition from an endpoint in a pair to a blocking node.

Lemma 3. *Let $X = \{X_1, X_2, \ldots, X_p\}$ be pairs of vertices inside the n-cube with $d(X_i) = n$ for $1 \leqslant i \leqslant p$, and $\{X_{p+1}, X_{p+2}, \ldots, X_{p+q}\}$ be a set of blocking nodes such that $1 \leqslant p \leqslant p + q \leqslant n$, $p < n$ and $n > 3$. There exist at least $n - p - q + 2$ bit positions k such that the cardinality of the set*

$$e(X) \bigcup \{s_i \oplus 2^k \mid 1 \leqslant i \leqslant p\} \bigcup \{t_i \oplus 2^k \mid 1 \leqslant i \leqslant p\}$$

is $4p + q - 1$ (conflict or blocked case) or $4p + q$ (conflict-free or unblocked case). This means that for each of these bit k positions there is at most one transition that causes a conflict. Furthermore the conflict is incident on a blocking node: for fixed k it occurs due to one unique $\ell \in \{1, \ldots, p\}$ and either $s_\ell \oplus 2^k$, or $t_\ell \oplus 2^k$, but not both, is a conflicting blocking node.

Proof. By Lemma 2 we know that if $p + q < n$, then there exists $n - p - q + 1$ bit positions k such that

$$e(X) \bigcup \{s_i \oplus 2^k \mid 1 \leqslant i \leqslant p\} \bigcup \{t_i \oplus 2^k \mid 1 \leqslant i \leqslant p\}$$

is of cardinality $4p + q$. So let us consider now the case when $p + q = n$. Since $p < n$ then $q \geqslant 1$. Suppose that we delete one of the blocking nodes, z. Then Lemma 2 holds and we know that $n - p - q + 2$ conflict-free bits exist. Let us now consider each of the $n - p - q + 2$ conflict-free bits just identified. Let k be one such bit. When the blocking node z is added back, then it remains conflict-free or it introduces conflicts. If node z causes a conflict along bit k, then for some ℓ, $s_\ell \oplus 2^k = z$, or $t_\ell \oplus 2^k = z$. Note that if both are true, then $s_\ell = t_\ell$ which is precluded. Without loss of generality assume $s_\ell \oplus 2^k = z$. We claim that no other $\ell' \neq \ell$ exists such that $s_\ell \oplus 2^k = s_{\ell'} \oplus 2^k$, or $s_\ell \oplus 2^k = t_{\ell'} \oplus 2^k$, since either case implies that $s_\ell = s_{\ell'}$, or $s_\ell = t_{\ell'}$ which contradicts the problem definition. Therefore s_ℓ is unique and there is one conflict for bit k. The cardinality of set $e(X)$ for bit k is either $4p + q - 1$, or $4p + q$. The lemma follows from the fact that the above statement holds for all the $n - p - q + 2$ bits k identified when we deleted blocking node z. □

Before we establish our main result, we explain in detail the q-dependent transformation. In the proof of our theorem we apply this transformation when $p \geqslant q$. In the q-dependent transformation, as in the 1-bit step, we start by selecting the bit k which we know exists by Lemma 3 since $p + q \leqslant n$ and $p < n$. First let us discuss the case where there is a bit k that is conflict-free. Let i be the number of q blocking nodes with a one in the kth bit position. Without loss of generality one may assume that $i \leqslant \lfloor q/2 \rfloor$. Because if this is not the case all the vertices may be complemented with $2^n - 1$ and the new value for i will be at most $\lfloor q/2 \rfloor$.

The transformation makes i transitions from an endpoint of p pairs from 1 to 0 along the kth bit and $p - i$ transitions from an endpoint for all of the remaining pairs from 0 to 1. Therefore, the subproblem in which all the kth bit positions are 1 will have the $p - i$ pairs that made the initial transition from 0 to 1, plus the i blocking nodes with a one in the kth bit position plus one endpoint of the i pairs that made the transition from 1 to 0. The resulting problem is an instance of $(n, p - i, 2i)$. The other subproblem is one in which all the kth bit positions are 0s and will have the i pairs that made the transition from 1 to 0, plus the $q - i$ blocking nodes with a zero on the kth bit position plus one endpoint of the $p - i$ pairs that made the transition from 0 to 1. This resulting problem is an instance of $(n, i, p + q - 2i)$.

Now let us consider the case when each possible bit k given by Lemma 3 has a conflict with a blocking node. By definition when $i = 0$ all the q blocking nodes have a zero in the k-bit position. Therefore the transition from a 0 to a 1 along bit k in a pair will not have a conflict with a blocking node. Since all pairs make this type of transition, it follows that our transformation rules are valid. When $i > 0$ one may always select pairs for making the transition from 1 to 0 on bit k that do not have the conflict and such a pair exists because at most $p/2$ pairs make the transition from 1 to 0. Therefore the above q-dependent transformation is valid in all cases. The two most important properties of the q-dependent transformation are avoiding the conflict when making the transition on bit k, and making the resulting subproblems fall within the induction hypothesis.

Theorem 3. *Given X consisting of p pairs of nodes and q blocking nodes in an n-cube. If for all i and j such that* $1 \leqslant i \leqslant p < j \leqslant p + q$, $d(X_i) = n > 6$, $d(X_j) = 0$, $p + q \leqslant n - 1$, *then node disjoint shortest paths exist for X.*

Proof. The base case of an induction proof is given by proving that the theorem holds for $n \in \{7,8\}$. In the case of $n = 7$ the initial problem is an instance of $(7,6,0)$ and through the application of the balanced transformation becomes two problems which are instances of $(n',p',q') = (6,3,3)$. Each of these subproblems satisfies the conditions of Lemma 3, therefore a bit k for making the transition exists, but it may have one conflict. As one may complement the vertices in the problem by $2^7 - 1 = 111111$; the number of blocking nodes, i, with a 1 in the kth bit position is in the range $0 \leqslant i \leqslant \lfloor 3/2 \rfloor = 1$. There are thus two different cases for which we apply the q-dependent transformation. Table 1 lists the resulting instances for the two possible values of i. Note that we can always perform the transition while avoiding the conflict identified in Lemma 2. The first column represents the instance in the subcube in which the k-bit is 1 and the second column represents the one in which the k-bit is zero.

It is simple to see that every instance of problems $(5,3,0)$, $(5,2,2)$ and $(5,1,4)$ falls into the conditions of Theorem 1. Every problem instance of $(5,0,6)$ has a solution simply because there are no pairs, just blocking nodes. Therefore disjoint paths exist for the instances of $(6,3,3)$ and therefore they exist for the instance of $(7,6,0)$.

Now consider any instance of $(8,7,0)$. By applying the balanced transformation we only need to show that the resulting instances of $(7,4,3)$ have a solution. Since the conditions of Lemma 3 apply, there exists a bit k for making a transition that may have a conflict. As mentioned before, we may assume that the number of blocking nodes with a 1 in the k bit positions is $0 \leqslant i \leqslant \lfloor q/2 \rfloor = \lfloor 3/2 \rfloor = 1$. Thus, applying the general q-dependent transformation yields the instances given in Table 2. Note that we can always perform the transition while avoiding the conflict identified in Lemma 2.

Every instance of $(6,0,7)$ has a solution because there are only blocking nodes. Every instance of $(6,1,5)$ falls within the conditions of Lemma 1 and therefore a

Table 1
Resulting Problem instances for (6,3,3)

i	Transitions	
	0→1	1→0
0	(5,3,0)	(5,0,6)
1	(5,2,2)	(5,1,4)

Table 2
Resulting problem instances for (7,4,3)

i	Transitions	
	0→1	1→0
0	(6,4,0)	(6,0,7)
1	(6,3,2)	(6,1,5)

shortest path exists for the pairs. As established above every instance of $(6,3,3)$ has a solution, therefore every instance of $(6,3,2)$ has also a solution since one can simply add a dummy blocking node. The instance of $(6,4,0)$ requires further reduction using the balanced transformation. For this problem we can apply Lemma 2, so it is possible to make a conflict-free transition on bit k. One may thus apply the balanced transformation to any instance of $(6,4,0)$ yielding two instances of $(5,2,2)$. These problems have a solution due to Theorem 1. This completes the proof of the base case.

Now let us prove the induction hypothesis. Assume that the theorem holds for every instance in the $(n-1)$-cube and prove it for any instance in the n-cube. One only needs to consider the case wherein $q=0$, since all blocking nodes can be transformed into pairs. The resulting problem is therefore an instance of $(n,p,0)$ for $p<n$.

Lemma 2 establishes that at least one bit for making a transition exists. Let k be that bit position. By making a transition on half of the pairs along bit k from 0 to 1 and the remaining pairs from 1 to 0 the two resulting subproblems are instances of

$$\left(n-1,\left\lfloor\frac{p}{2}\right\rfloor,\left\lceil\frac{p}{2}\right\rceil\right) \quad \text{and} \quad \left(n-1,\left\lceil\frac{p}{2}\right\rceil,\left\lfloor\frac{p}{2}\right\rfloor\right).$$

If a solution exists for every instance of $(n-1,\lceil p/2\rceil,\lfloor p/2\rfloor)$, then one also exists for every instance of $(n-1,\lfloor p/2\rfloor,\lceil p/2\rceil)$ because one may always transform a blocking node into a pair. Therefore we only need to prove that a solution to every instance of $(n',p',q')=(n-1,\lceil p/2\rceil,\lfloor p/2\rfloor)$ exists. Note that $p'\geqslant q'$ and therefore the q-dependent transformation may be applied.

When $p'+q'<n'=n-1$ the induction hypothesis can be applied, so the remaining case is when $p=p'+q'=n'=n-1$. In this case we know by Lemma 3 that a bit k for further reduction exists. However, note there may be at most one conflict incident on a blocking node as established in Lemma 3. Let i be the number of individual nodes out of the q' blocking nodes with a one in the kth bit position. If $i>\lfloor q'/2\rfloor$, then complement ($\oplus$) the subproblem input with $2^n-1=\underbrace{11\cdots1}_{n'}$. Thus $0\leqslant i\leqslant\lfloor q'/2\rfloor$ holds.

Even if a bit k is blocked per Lemma 3 one may apply the q-dependent transformation to the instance of (n',p',q') to yield two distinct instances of

$$(n'',p'',q'')=(n'-1,p'-i,2i) \quad (0\to 1 \text{ transition})$$

and

$$(n''',p''',q''')=(n'-1,i,q'+p'-2i) \quad (1\to 0 \text{ transition}).$$

The reason why it is possible to apply these two transformations is that we can always select the pair with the conflict so that it makes the appropriate transition and avoid the conflict. When i is equal to zero we make all the transitions for the pairs from a zero to a one, but since i is zero all the k bits of the blocking nodes are one. For the other values of i one can always place the pair with the conflict to make the appropriate transition.

The former instance (n'', p'', q'') is within the induction hypothesis when $p'' + q'' \leqslant n'' - 1$. The sum of p'' and q'' is equal to

$$p' - i + 2i = p' + i = \left\lceil \frac{p}{2} \right\rceil + i.$$

This expression is maximum when i has the maximum value which is $\lfloor q'/2 \rfloor$. Therefore,

$$p'' + q'' \leqslant \left\lceil \frac{p}{2} \right\rceil + \left\lfloor \frac{q'}{2} \right\rfloor = \left\lceil \frac{p}{2} \right\rceil + \left\lfloor \frac{\left\lfloor \frac{p}{2} \right\rfloor}{2} \right\rfloor \leqslant n - 3,$$

since $n > 6$ and the instance of (n'', p'', q'') falls within the induction hypothesis.

Instance (n''', p''', q''') falls within the induction hypothesis when $p''' + q''' = q' + p' - i = p - i \leqslant n - 3$. This holds true when $i \geqslant 2$. So let us now consider the remaining cases, $i \in \{0,1\}$. When $i = 1$, Lemma 1 establishes that every instance of $(n-2, 1, p-2)$ has a solution when $p - 2 < n - 2$. This is always the case as $p < n$. When $i = 0$ one notes that every instance of $(n-2, 0, q' + p')$ has a solution since there are only blocking nodes. □

By examining the proof of Theorem 3 it is clear that it is easier to establish correctness for the first algorithm (Theorem 1). In the next subsection we present our algorithm whose proof of correctness is relies upon Theorem 3.

3.1. Algorithm for the balanced approach

An algorithm based on a straightforward implementation of Theorem 3 takes exponential time. Let us explain why this is the case with an example. Consider an instance of $(64, 63, 0)$. The algorithm generates an instance of $(63, 32, 31)$ and one of $(63, 31, 32)$. The instance of $(63, 31, 32)$ reduces to $(63, 32, 31)$ by transforming a blocking node into a pair. Further reduction of the instance of $(63, 32, 31)$ using the conflict-free part of Lemma 3 and applying the q-dependent transformation yields an instance of $(62, 32-i, 2i)$ and one of instance $(62, i, 63-2i)$. When $i = 2$ the resulting instances are $(62, 30, 4)$ and $(62, 2, 59)$. Now applying the induction hypothesis we transform the instance of $(62, 2, 59)$ into one of $(62, 61, 0)$. Therefore to solve the instance $(64, 63, 0)$ we need to solve two instances of $(62, 61, 0)$ plus we need to do some extra work. So the time complexity of the algorithm appears to be exponential, because we are converting blocking nodes into pairs per the simplification of the proof of Theorem 3. However, the algorithm does not really need to produce paths for the dummy pairs being introduced.

To reduce the time complexity bound we make the algorithm mimic the conversion of blocking nodes into dummy pairs. If there are no (original) pairs, then the algorithm is done; if there is only one pair, then to find the path it applies Lemma 1; and if there is more than one pair, it applies the constructive proof of Theorem 3 to generate the paths. When the dimension is 7 or 8, then it just applies the constructive proof for the base case of Theorem 3 to find the paths. The algorithm uses the proof of Theorem 3 in a two stage process to partition the problem. Our algorithm is initially invoked with Find_Paths_0(n,X).

```
Find_Paths_0(n, X = {X_1,X_2,...,X_p, X_{p+1},...,X_{p+q}})
{assume that d(X_1) = d(X_2) = ··· = d(X_p) = n,d(X_{p+1}) = d(X_{p+2}) = ··· = d(X_{p+q}) = 0,
     all the X_i pairs are in the same n dimensional subcube, and p + q ⩽ n − 1
   if p = 0 return;
   if p = 1 construct path using the proof of Lemma 1 and return;
   if n ⩽ 8 return the paths constructed by the base case in Theorem 3;
   k ←Find_Bit_2(X); // using the proof of Lemma 2 find an unobstructed bit
   X' ← OneToZeroBalancedTransition(k, X);// Transformation is defined below
   X'' ← ZeroToOneBalancedTransition(k, X); // Transformation is defined below
   Output_Paths(k, X);
   Find_Paths_1(n − 1,X');
   Find_Paths_1(n − 1,X'');
}
```

The bit k is conflict-free due to Lemma 2. The ZeroToOneBalancedTransition(k, X) and OneToZeroBalancedTransition(k, X) return the problem partitioned via the balanced transformation. The ZeroToOneBalancedTransition (k, X) takes all pairs and returns $X' = \{X'_1, X'_2, \ldots, X'_{\lceil p/2 \rceil}, X'_{\lceil p/2 \rceil + 1}, \ldots, X'_{p+q_0}\}$, $\lceil p/2 \rceil$pairs reduced in size by making a transition from a 0 to 1 on bit k. Endpoints of the remaining $\lfloor p/2 \rfloor$with a one in the kth bit position are returned as blocking nodes. We use q_0 to denote the number of singleton nodes with a one in the kth bit position. X'' = OneToZeroBalancedTransition (k, X) returns the complement, albeit returning $\lfloor p/2 \rfloor$ paths with $q - q_0 + \lceil p/2 \rceil$ singleton nodes. While the proof of Theorem 3 is simplified by converting blocking nodes to paths, the algorithm keeps actual pairs distinct from blocking nodes.

```
Find_Paths_1(n,X = {X_1,X_2,...,X_p,X_{p+1},...,X_{p+q}})
assume that d(X_1) = d(X_2) = ··· = d(X_p) = n,d(X_{p+1}) = d(X_{p+2}) = ··· = d(X_{p+q}) = 0,
all the X_i pairs are in the same n dimensional subcube, and p + q ⩽ n
if p = 0 return;
if p = 1 construct path per Lemma 1, output path and return;
if n ⩽ 8 return the paths constructed by the base case in Theorem 3;
if p + q < n then Find_Paths_0(X);//Falls within the inductive hypothesis of
   Lemma 2.
else {// p + q = n
   k ← Find_Bit_3(X); //Using the proof of Lemma 3 find an appropriate bit.
   // If a conflict-free bit exists select it.
   //define CountBlockingNodesWithOne(k, X) as the number of
   // blocking nodes with a 1-bit in the kth position.
   if (CountBlockingNodesWithOne(k, X) > ⌊q/2⌋) then
   X = ComplementProblem(n,X);
   // the problem must be complemented appropriately upon output.
   i ← CountBlockingNodesWithOne(k, X);
   X' ← OneToZero_qTransition(i, k, X); //Transformation is defined below
   X'' ← ZeroToOne_qTransition(i, k, X); //Transformation is defined below
```

```
    Output_Paths(k, X);
    Find_Paths_0(n − 1, X′);
    Find_Paths_0(n − 1, X″);
}
```

Function CountBlockingNodesWithOne(k, X) returns the number of blocking nodes with a one in the kth bit position. ComplementProblem(n, X) complements all nodes with $2^n - 1$ while maintaining the pair and blocking node structure. Note that under complementation the procedure Output_Paths must complement its output. To simplify the presentation of the algorithm we omit the details.

Both OneToZero_qTransition(i, k, X) and ZeroToOne_qTransition(i, k, X) are based on the q-dependent transformation. As mentioned prior to Lemma 3, the transformation can be applied whether or not there is a conflict.

The function ZeroToOne_qTransition(i, k, X) does not have a conflict. This procedure is given p pairs, it selects $p - i$ of them to make a transition from 0 to 1. The residual blocking nodes from pairs is i. However in the event that $i = 0$ and there is a conflict with a blocking node as indicated in Lemma 3 there are no conflicts because all the blocking nodes have their k bit equal to 0, but we make all transitions from a 0 to a 1. When $i > 0$ one can always select the pair with the possible conflict to make the the appropriate transition that will avoid the conflict.

Similarly the function OneToZero_qTransition(i, k, X) returns i pairs which are capable of making the transition from 1 to 0 on bit k. If there is a conflict, as in Lemma 3, then when $i > 0$ one simply selects i alternate pairs for making the transition from 0 to 1. The number of residual blocking nodes with a one in the kth bit position placed into the input to this function. When $i = 0$ we do not use this type of transition.

Theorem 4. *Given X consisting of p pairs of nodes and q blocking nodes in an n-cube. If for all i and j such that $1 \leqslant i \leqslant p < j \leqslant p + q$, $d(X_i) = n > 6$, $d(X_j) = 0$, $p + q \leqslant n - 1$, then algorithm Finds_Paths_0 constructs node disjoint shortest paths for X and can be implemented to take $O((p^2 + pq)n^3)$ time.*

Proof. The proof of correctness follows directly from the constructive proof of Theorem 3 and the fact that our algorithm just mimics the conversion of blocking nodes into pairs.

All the steps can be implemented to take $O((p + q)n)$ time except for procedures Find_Bit$_2$ and Find_Bit$_3$; and constructing a path with Lemma 1. The construction of the path with Lemma 1 takes $O(n^2)$ time. Procedures Find_Bit$_2$ and Find_Bit$_3$ are implemented using a binary trie for set insertion. The former invocation allows no conflicts for the $4p + q$ insert operations. The latter call allows at most one conflicting insert operation and again represents $4p + q$ insert operations. Each endpoint and blocking node is n bits long and the number of insert operations is $4p + q$. Therefore, this operation will take at most $O(n^2(p + q))$ steps.

Because of the recursive structure of the program one may (incorrectly) think that the time complexity bound is exponential. However, the number of calls is

significantly limited at each level. The condition of Theorem 3 requires that $p < n$. For each value of n there are at most p invocations that have at least one pair. Thus $O(p)$ problems are active at each of the n levels. Therefore the total number of calls is $O(pn)$.

Since the sum of the values of $p + q$ at each invocation is not larger than the sum at the previous step it then follows that, the algorithm runs in $O(p\ n(n^2(p + q))) = O((p^2 + p\ q)n^3)$. □

Since the number of input bits is $m = (2p + q)n$, the overall time complexity is $O(m^3)$.

4. Extreme version of the edge disjoint shortest paths problem

In this section we discuss the extreme version of the *edge* disjoint shortest paths problem in the *undirected* n-cube where again $d(X_i) = n$. We present an efficient algorithm for the case where $n > 0$ and n odd. First we prove that every problem instance has a solution and then we present an efficient algorithm to construct a set of such paths.

Assume without loss of generality that for each pair X_i, s_i has the leading bit equal to zero and t_i has the leading bit equal to 1 or equivalently $s_i < t_i$. In Theorem 5 we prove that every k pairwise edge disjoint shortest path problem has a solution when $k = 2^{n-1}$. As any subset of pairs may be deleted the same results holds when $k < 2^{n-1}$ by simply removing the paths for such pairs. Assume without loss of generality that the ith pair has source s_i which is labeled with the bit representation of i including leading zeros.

Theorem 5. *Given any partial half routing request in an undirected n-cube with $p = 2^{n-1}$ pairs and $d(X_i) = n$, edge disjoint shortest paths exist when $n > 0$ and n odd.*

Proof. The base case $n = 1$ trivially has an edge disjoint shortest path $0 \leftrightarrow 1$. Assume that there is a set of edge disjoint shortest paths for $k = 2^{n-3}$ pairs for all problem instances with $n - 2 \geqslant 1$ and n odd. Now let us prove that edge disjoint shortest paths exist for n.

Let us divide the n-cube into four $(n - 2)$-cubes along dimension 0 and 1. We refer to these subcubes as the 00, 01, 10 and 11 cubes depending on the value of bit 1 and bit 0. Clearly each pair has its endpoints in subcubes 00 and 11, or in 01 and 10. The idea is to use all the edges between the four cubes to route the pairs starting either at s_i or t_i for each i so that all pairs end up with both endpoints in the same subcube, each vertex has exactly one endpoint of a pair, and the edges between the subcubes are used once.

For each string A of $n - 2$ bits that starts with a zero there are exactly four pairs X_{A00}, X_{A01}, X_{A10} and X_{A11} such that their endpoints are located at the vertices whose first $n - 2$ bits in its bit address are equal to A or $\overline{A}$. The pairs have bit addresses $(A00, \overline{A}11)$, $(A01, \overline{A}10)$, $(A10, \overline{A}01)$, $(A11, \overline{A}00)$. The routing for these pairs is as follows. The paths make the transitions between the four subcubes of size $n - 2$.

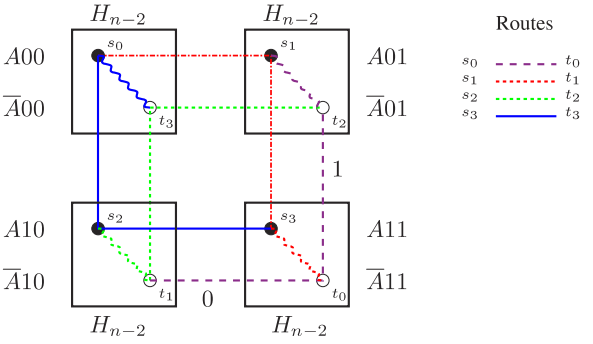

Fig. 3. Induction partition of the routes.

$$A00 \leftrightarrow A01 \leftrightarrow A11 \leftrightarrow \cdots \leftrightarrow \overline{A}11$$

$$A11 \leftrightarrow A10 \leftrightarrow A00 \leftrightarrow \cdots \leftrightarrow \overline{A}00$$

$$A10 \leftrightarrow \cdots \leftrightarrow \overline{A}10 \leftrightarrow \overline{A}00 \leftrightarrow \overline{A}01$$

$$A01 \leftrightarrow \cdots \leftrightarrow \overline{A}01 \leftrightarrow \overline{A}11 \leftrightarrow \overline{A}10$$

The resulting pairs are ($A11$, $\overline{A}11$), ($A00$, $\overline{A}00$), ($\overline{A}10$, $A10$) and ($\overline{A}01$, $A01$) each of which is in its own subcube.

Fig. 3 represents the case when $A = \underbrace{000\cdots0}_{n-2}$. Note that none of the above edges used in the transitions are used more than once and no vertex ends up with more than one endpoint. In addition in the resulting subproblem all the edges are between vertices whose first $n-2$ bit address is either A or $\overline{A}$. Therefore there is no interference with paths for different A's or different $\overline{A}$'s. Fig. 3 shows the routes.

So the resulting subproblems in each of the subcubes satisfies the induction hypothesis after deleting the rightmost two bits. The theorem follows by induction. □

As mentioned before by deleting any subset of routes one can use Theorem 5 to establish Corollary 1.

Corollary 1. *Given any partial half routing request in an undirected n-cube with* $p \leqslant 2^{n-1}$ *pairs and* $d(X_i) = n$ *edge disjoint shortest paths exist when* $n > 0$ *and n odd.*

The following program finds edge disjoint shortest paths for all n and p when n is odd. The input is the routing request X and the dimension of the n cube and the output is a list of dimensions for each pair $X_j = (s_j, t_j)$ representing the transition made in the path from s_j to t_j. The algorithm considers each pair at a time. For each pair X_j the program follows the proof of Theorem 5. When making the transition from s_j they are printed directly, but when they are for t_j they are pushed onto a stack and outputted in the reverse order so they correspond to the path from s_j to t_j. The program assumes that $s_j < t_j$. X is a partial half routing

request with all $d(X_j) = n$. *Stack* is a stack initialized by $Stack \leftarrow \perp$, tested for emptiness with the expression ($Stack = \perp$) and with the usual operations Stack. Push(a) and Stack.Pop(). As the input size of the problem is O($p\ n$) it is a linear time algorithm.

```
Find_Paths(n, X = {X_1, X_2, ..., X_p}) {
  for j ← 1 to p {
    (s_j, t_j) ← X_j
    Stack ← ⊥
    for k ← 0 to ⌊(n−1)/2⌋ {
     b0 ← value of bit 2k of s_j
     b1 ← value of bit 2k + 1 of s_j
     case ((b1,b0)) {
                (1,1): Output(2k); Output(2k + 1);
                 break;
                (0,1): Stack.Push(2k + 1); Stack.Push(2k);
                 break;
                (0,0): Output(2k + 1); Output(2k);
                 break;
                (1,0): Stack.Push(2k); Stack.Push(2k + 1);
                 break;
                }
     }
    while (Stack ≠ ⊥)
     Output (Stack.Pop());
  }
}
```

Theorem 6. *For $n > 0$ and n odd in an undirected n-cube with $p \leqslant 2^{n-1}$ pairs and $d(X_i) = n$ edge disjoint shortest paths the Algorithm Find_Paths above correctly establishes and outputs edge disjoint shortest paths with time complexity $O(p\ n)$ or linear with respect to number of input bits.*

Proof. The outer loop is iterated p times and the inner loop n times. The inner most body of the loop is considered to take constant time to execute. Therefore the total time complexity for this algorithm is O(pn). Correctness follows directly from the fact that the algorithm parallels the proof of Theorem 5 in an iterative manner. □

In the case of Theorem 5 there are 2^{n-1} pairs of length n. When n is even, not all 2^{n-1} distance n pairs have edge disjoint shortest paths. Clearly when $n = 2$ and $X = \{(00,11),(01,10)\}$ the routing request does not have an edge disjoint route. We now show that there is no edge disjoint shortest paths routing in an undirected n cube with 2^{n-1} pairs of length $n > 2$. The hypercube has 2^n nodes and $n2^{n-1}$

undirected edges and each path for the pair requires n such edges. Therefore in order to have edge disjoint paths all edges must be used in by the paths. Consider any node which is a source in a path, denote the source s_0. One edge must emanate from s_0 in the path to node t_0. Therefore the number of remaining (unused) edges is odd. However every path that goes through s_0 must use exactly two edges. Therefore any set of edge disjoint paths uses an odd number of the edges emanating from s_0. As we have an even dimension n it must be that at least one edge emanating out of s_0 is not used. This implies that all edge disjoint shortest paths problems in the undirected n-cube for 2^{n-1} pairs do not have a solution for n even.

5. Future work

There are instances of (n,p,q) with $p + q = n$ that do not have node disjoint shortest paths. So it is not possible to extend Theorem 4 to $p + q = n$. An interesting area for future research is to find improved sufficiency conditions and perhaps even necessary and sufficient conditions for the existence of node disjoint shortest paths in the n-cube. One caveat however, determining whether or not node disjoint shortest paths exist is an NP-complete problem in an n-cube [3,4]. Therefore, necessary and sufficient conditions for the existence of node disjoint shortest paths are most likely not polynomial time computable. However, the extreme version of the problem with an arbitrary number of pairs is not known to be NP-complete.

Another very interesting open problem is to develop an algorithm to determine in polynomial time whether for any partial half permutation routing request, edge disjoint shortest paths for m pairs in an undirected $2n$ cube exist. Clearly the answer is "no" to this decision problem for $m = 2^{n-1}$ pairs of length n when n is even, but for many other problem instances a solution does exist.

References

[1] S. Gao, B. Novick, K. Qui, From Hall's matching theorem to optimal routing on hypercubes, Journal of Combinatorial Theory Series B 74 (2) (1998) 291–301.

[2] T.F. Gonzalez, F.D. Serena, Node disjoint shortest paths for pairs of vertices in an n-cube network, in: Proceedings of the International Conference on Parallel and Distributed Computing and Systems (PDCS2001) (2001), IASTED, pp. 278–282.

[3] T.F. Gonzalez, F.D. Serena, Complexity of k-pairwise disjoint shortest paths in the hypercube and grid networks. Technical Report TRCS-2002-14, University of California at Santa Barbara, May 2002.

[4] T.F. Gonzalez, F.D. Serena, Complexity of k-pairwise disjoint shortest paths in the undirected hypercubic network and related problems, in: Proceedings of the International Conference on Parallel and Distributed Computing and Systems (PDCS 2002), IASTED, 2002.

[5] T.F. Gonzalez, F.D. Serena, n-Cube search algorithm for finding $(n-1)$-pairwise node disjoint shortest paths, in: International Conference on Communications in Computing (CIC 2002), CSREA Press, 2002.

[6] Q.-P. Gu, S. Peng, k-Pairwise cluster fault tolerant routing in hypercubes, IEEE Transactions on Computers 46 (1997) 9.

[7] Q.-P. Gu, S. Peng, Node-to-set and set-to-set cluster fault tolerant routing in hypercubes, Parallel Computing 24 (1998) 1245–1261.
[8] E. Horowitz, S. Sahni, Fundamentals of Data Structures, W.H. Freeman and Company, 1990.
[9] R. Karp, On the computational complexity of combinatorial problems, Networks 5 (1975) 45–68.
[10] S. Latifi, H. Ko, P.K. Srimani, Node-to-set vertex disjoint paths in hypercube networks. Computer Science Technical Report, Colorado State University CS-98-107, 1998.
[11] S. Madhavapeddy, I.H. Sudborough, A topological property of hypercubes: node disjoint paths, in: Proceedings of Second IEEE Symposium on Parallel and Distributed Processing, 1990, pp. 532–539.
[12] S. Madhavapeddy, I.H. Sudborough, Disjoint paths in the hypercube, in: M. Nagl (Ed.) WG (June 1990), Lecture Notes in Computer Science. Graph-Theoretic Concepts in Computer Science, 15th International Workshop, WG'89, vol. 411, pp. 3–18.
[13] K. Menger, Zur allgemeinene kurventheorie, Fundamental Mathematics 10 (1927) 95–115.
[14] M.O. Rabin, Efficient dispersal of information for security, load balancing and fault tolerance, Journal of the Association of Computing Machinery 36 (2) (1989) 335–348.
[15] Y. Shiloach, Two paths problem is polynomial, Technical Report TR-CS-78-654, Stanford University, 1978.
[16] M. Watkin, Graph is $(2k-1)$-connected is a necessary condition to admit k paths, Duke Mathematical Journal 23 (1968).