n-Cube Network: Node Disjoint Shortest Paths for Pairs of Vertices

Teofilo F. Gonzalez and F. David Serena
{teo, dserena}@cs.ucsb.edu
University of California at Santa Barbara
Department of Computer Science

1 Abstract

In parallel and distributed systems many communications take place concurrently, so the routing algorithm as well as the underlying interconnection network plays 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 an efficient algorithm for the n-cube pairwise node disjoint shortest paths problem in the presence of faulty nodes. The time complexity of the algorithm is $O(m^{2.5})$, when the input length is $m = O(n^2)$.

Keywords: fault-tolerance, hypercube, n-cube, pairwise node disjoint shortest paths, routing, k-pairwise.

2 Introduction

The n-cube is a vital structure for parallel computing. Several systems with this communication architecture have been built in the past and the SGI Origin 2000 [2] is a recent computing system 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 an efficient algorithm for a routing problem which is common to many applications.

Our node disjoint shortest paths problem is defined over the n-cube and there are p pairs of vertices $X = \{X_1, X_2, \ldots, X_p\}$, where $X_i = \{s_i, t_i\}$ specified as part of the input. The problem consists of constructing *node disjoint shortest paths* from s_i to t_i for all $1 \leq i \leq p$. Zero length pairs or blocking nodes may also be included in the problem input. We show that under certain conditions a solution exists and we present an $O(m^{2.5})$ time algorithm, where m is the input length, to construct a set of such paths.

Node disjoint paths have been studied for many years. Karp [9] showed that determining whether or not there exist node disjoint paths for a set of pair of points in a graph is an NP-complete problem. Shiloach [15] presented a polynomial time algorithm to construct node disjoint paths in a graph for two pair of points 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.

Madhavapeddy and Sudborough [12] developed an $O(n^3 \log n)$ algorithm to find disjoint paths for k pairs in an n-cube when $k > 2$ and $n \geq 4$. Each of the paths is of length at most $2n$. Then Gu and Peng [7] 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, when $k \leq \lceil n/2 \rceil$. In VLSI routing the problem finding disjoint paths in 1 and 2 dimensions has been extensively studied [1, 11]. For brevity we do not discuss these results in detail.

The main difference between the work mentioned above and ours is that we are interested in finding node disjoint **shortest** paths rather than just node disjoint paths. Rabin's [13] proposed algorithms in which an attempt is made to simultaneously analyze fault-tolerance, security and load balancing. While the paper is an interesting starting point note that in general, three arbitrary parameters of optimization may sometimes be contradictory or at least mutually interdependent.

Gu and Peng [8] address an interesting and very similar problem; 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 every vertex in one set to a different vertex in the other set; where as, in our problem one constructs **shortest** paths for a **given set of pair** of vertices. 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. Gu and Peng's paper [8] also study the node-to-set node disjoint paths problem. Gao, Novick and Qiu [4] presented an algorithm for finding node-to-set node disjoint *shortest* paths. This problem reduces to our problem after finding the first link for each path. Gao, Novick and Qiu's paper [4] exploits the fact that local constraints, the existence of a first step in a path, dominate in determining the existence of node-to-set node disjoint paths. Namely, as shown in [4], the existence of an SDR or System of Distinct Representatives 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. Node-to-set approaches in the n-cube are highly pragmatic in the sense that it has applications in the context of fault-tolerant distributed networks. There are several important papers which address this issue for the n-cube, [4, 10]. Finding k disjoint paths between two nodes in the hypercube has also been studied [7], but for brevity we do not discuss in detail these results.

In this paper we address the problem of finding node disjoint shortest paths in an n-cube. How difficult is this problem? Karp showed that determining whether or not node disjoint paths exist in a graph is an NP-complete problem [9]. By using his same reduction but starting with a restricted version of SAT, one can show that the problem remains NP-complete even if one restricts all the node disjoint shortest paths to be of length at most 3 [5]. The same reduction does not seem to be extendible to the n-cube. Gonzalez and Serena [5] have also shown, by using a different approach, that finding k node disjoint shortest paths in an n-cube is NP-complete even when the length of each of the paths is at most three, but polynomially solvable, even on general graphs, when the length of each path is at most two. For this case we know that the problem is computationally equivalent to finding a maximum matching in a bipartite graph [5]. The problem of finding node disjoint shortest path problem in a mesh when one restricts all paths being of length at most 3 is polynomially solvable [5].

In the next subsection we discuss the topology of the n-cube and then in the next section we show that node disjoint shortest paths in the n-cube exist for a large set of problem instances. Then we present our algorithm, based on our constructive proofs, to generate one set of such paths. The time complexity of our algorithm is $O(m^{2.5})$, when m is $O(n^2)$.

2.1 Topology of an n-Cube

An n-cube, or equivalently an n dimensional hypercube, is an undirected grap h $G = (V, E)$ with vertices $V = \{0, 1, \ldots, 2^n - 1\}$ represented by string of bits in the range

$$\underbrace{00\ldots0}_{n}{}_2 = 0 \leq v \leq 2^n - 1 = \underbrace{11\ldots1}_{n}{}_2 \,.$$

The undirected edges E are given by the set $\{\{a, b\} \mid d(a, b) = 1\}$ where $d(a, b)$ is the number of bits with value one in the binary representation of $a \oplus b$ and $\oplus$ is the bitwise "exclusive or" operation.

This representation of the n-cube is prevalent in the literature of coding theory with the work of Frank Gray and the "Gray Codes". It is well known that paths in an n-cube may be represented with Gray codes. In fact other combinatorial objects such as permutations can be made to have Gray-Code-like orderings [14]. Also, the n-cube may be viewed as the lattice of the power set of the set $\{0, 1, 2, \ldots, n-1\}$ ordered by set inclusion [3].

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

$$\Theta(s, t, u) = (s \oplus t == s \oplus u \text{ “bitwise or” } u \oplus t).$$

An alternate predicate, equivalent for the n-cube, is the distance based predicate

$$\Theta(s, t, u) = (d(s, u) + d(u, t) == d(s, t)),$$

where $d(s, t)$ is the number of edges between node s and t in the n-cube.

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 by 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 [4] where ordered sets are used to denote the transition from a node to the next in a shortest path. For example the path

$$000_2 \rightarrow 010_2 \rightarrow 011_2 \rightarrow 111_2$$

is denoted by the ordered set $O = (1, 2, 0)$. There are other equivalent notations in the literature, which we omit for brevity.

3 Node Disjoint Shortest Paths Problem

Let $X = \{X_1, X_2, \ldots, X_p\}$, where $X_i = \{s_i, t_i\}$, be a set of p pairs of n-cube vertices all of which are distinct. The *node disjoint shortest paths problem* is given X in the n-cube, where

n is the dimension of the cube, find node disjoint shortest paths connecting each s_i to t_i such that no two paths have a node in common. Each X_i is called a *pair* and the two nodes in it are called *endpoints* of X_i.

An extreme version of this problem is when $d(X_i) = n$, for all $1 \leq i \leq p$. The endpoints of every pair X_i are complements ($\oplus(2^n - 1)$) of each other because they are in the n-cube and $d(X_i) = n$, i.e., $\bar{s_i} = s_i \oplus (2^n - 1) = t_i$ We show that under certain conditions a set of shortest paths exists for the set of p pairs in X even in the presence of q blocking nodes, single vertices which can be considered as faulty or simply *blocking nodes*. Blocking nodes can be viewed as zero length shortest paths from a vertex to itself.

Our algorithm constructs the set of node disjoint shortest paths by finding for each pair i one link incident to an endpoint of X_i which will be part of the final path and then reduces the problem of finding p shortest paths in the presence of q blocking nodes in an n-cube into two independent problems in $(n-1)$-cubes with a total number of p pairs and $p+q$ blocking nodes. Our approach is to provide constructive proofs for our claims and then present an algorithm based on our constructive proofs. These proofs appear in Subsection 3.1 and 3.2, and the algorithm is given in Subsection 3.3. Latifi, Ko and Srimani [10] use a variation of this approach, albeit for a different routing problem, for which they generate node disjoint paths that are not necessarily shortest paths. The algorithm implied by the proof of Theorem 2 in Subsection 3.2 finds a link for each pair and reduces the problem, using a partitioning transformation, to finding a set of node disjoint paths in an $(n-1)$-cube. Then another transformation is applied using the special bit position shown to exist by Lemma 3 which further reduces the cube size to an $(n-2)$-cube. These two transformations conducted multiple times yields an algorithm which generates paths in a time efficient manner.

The sub-cube defined by the source and destination nodes s and t is defined by

$$K(s,t) = \{u \mid \Theta(s,t,u)\}.$$

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

3.1 The Simple Approach: $p \leq \lceil n/2 \rceil$ and $2p + q \leq n + 1$

Our problem 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 $X_i = \{s_i, t_i\}$, $d(X_i) = n$, for $1 \leq i \leq p$, and $X_i = \{a_i\}$ for $p+1 \leq i \leq p+q$, find node disjoint shortest paths for the pairs X_i that do not include blocking nodes and such that no two such paths have a node in common. We assume that all the endpoints and blocking nodes are different, i.e., the cardinality of the set $e(X)$ is $2p+q$, where

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

In this section we prove that when $p \leq \lceil n/2 \rceil$ and $2p + q \leq n+1$ node disjoint shortest paths exist. Our constructive proofs can be easily implemented to take $O(m^2)$, where $m = O(n^2)$ is the input length [6].

Our algorithmic approach, based on the proof of Theorem 1, is to construct the set of node disjoint shortest paths as follows. First we find an integer k which represents the position of one of the bits in the binary representation of the vertices and satisfies the property that for each endpoint $x \in X_i$ in every pair X_i, $1 \leq i \leq p$, 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 in the presence of q blocking nodes to two independent problems. One subproblem, which is rather simple, is in the sub-cube where bit k is zero and the other one in the sub-cube 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 transitioning on an endpoint of one pair using bit k. Lemma 1 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 inductively.

Let us illustrate our approach with the following example. The value of n is 3, $X_1 = \{000_2, 111_2\}$, and $X_2 = \{100_2, 011_2\}$. Transitioning on the bit $k = 0$ is not allowed because the neighbor of 000_2 along bit 0 is $100_2 \in e(X)$, but bit 1 and bit 2 are possible choices for k. Using bit $k = 1$ our approach is to select from X_1 the endpoint $e_1 = 000_2$ and from X_2 the endpoint $e_2 = 011_2$. Their corresponding neighbors are $g(e_1) = 010_2$ and $g(e_2) = 001_2$. Now the problem is to find a shortest path from 010_2 to 111_2 and one from 001_2 to 100_2. 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 resulting problems as finishing up the path for X_1 in the sub-cube $K(010_2, 111_2)$ with blocking node 011_2 and finishing up the path for X_2 in the sub-cube $K(001_2, 100_2)$ with blocking node 000_2. By deleting bit k the resulting problems reduce to finding a path in $K(00_2, 11_2)$ with blocking node 01_2 and finding a path in $K(01_2, 10_2)$ with blocking node 00_2. Once we find these paths we add the bit just deleted as well as the transition introduced on bit k and we get the node disjoint shortest paths in the 3-cube.

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

Proof: We prove this lemma by induction for all $n \geq 1$. The base case, $n = 1$ trivially holds because $s_1 = 0_2$, $t_1 = 1_2$ and $q = 0$, so the path $0_2 \leftrightarrow 1_2$ 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 $n > 1$. Without loss of generality assume $X_1 = \{s_1, t_1\}$, $s_1 = 0$ 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 sub-cube $K(0, 2^n - 1)$ 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 sub-cube $K(2^k, 2^n - 1)$ contains at most $q - \alpha_{\ell'} \leq n-2$ nodes and the resulting problem reduces to finding a path for a pair of nodes $\{2^k, 2^n - 1\}$ in the sub-cube $K(2^k, 2^n - 1)$ 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, ..., 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 sub-cube 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 sub-cube $K(2^k, 2^n - 1)$ with at most $n-2$ blocking nodes. The proof follows by the induction hypothesis after eliminating bit k. []

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 $1 \le i \le p$, and $\{X_{p+1}, X_{p+2}, \ldots, X_{p+q}\}$ be a set of blocking nodes such that $1 \le p \le p+q < n$ and $n > 2$. There exist at least $n-p-q+1$ bit position k such that the cardinality of the set*

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

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 all the paths.

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

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

Note that since $n > 2$, we know that $i \ne j$. The meaning of these S_l sets is as follows. If $S_l \ne \emptyset$ 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 need to show that at least $n-p+1$ sets S_l must be empty when $p \le n-1$. The proof is by contradiction. Assume that for every l, the number of empty sets S_l is fewer than $n-p+1$.

For each l, $0 \le l < n$, such that $S_l \ne \emptyset$ 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 $l' \ne l$ simply because if there exists $x \in X_i$, $y \in X_j$ such that $x \oplus y = 2^l$ and $\bar{x} \oplus \bar{y} = 2^l$ then $d(x, \bar{y}) > 1$ and $d(y, \bar{x}) > 1$. This precludes $\{i,j\}$'s membership in any $S_{l'}$, $l \ne l'$.

Therefore, all the sets a_l are distinct. Define the graph G' with the vertex set $V' = \{1, 2, \ldots, p\}$ and edge set $E' = \{a_i \mid 0 \le i < n\}$. 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 $r \le p < n$. 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 composed of the endpoints of pair X_{h_i} n-cube (G) has p edges (or transitions). Since each transition must be on a different bit and $d(X_{h_1})$ is equal to n, it must then be that $p = n$. But $p < n$, a contradiction.

Clearly, 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 is $p-1$. Thus, the number of empty sets S_l is at least $n-p+1$. The lemma follows for the case when $q = 0$.

When $q \geq 1$ one may add an endpoint to each blocking node and transform the problem to one consisting only of pair of points. Therefore the fact that the lemma holds for $q = 0$ and $p = n-1$ implies that it holds for $1 \leq p \leq p+q < n$. The constraint $n-p-q+1$ follows immediately because when we delete the added node we end up with the original problem instance and the conflicts in those nodes, if any, will be eliminated. blocking nodes as pair of points at a distance n from each other. This completes our proof of the Lemma 2. []

Our proof of Theorem 1 and algorithmic approach are based on the *1-bit step* for finding disjoint paths for p pairs and q blocking nodes in the n-cube. First we find using Lemma 2 an integer k that represents the position of one of the bits; such that, for every pair there is a shortest path that on one end makes a transition on bit k without a conflict.

The reduction uses a transition for one pair from 0 to 1 along the k^{th} bit and $p-1$ transitions from 1 to 0 for the remaining pairs. Therefore, the subproblem in which all the k^{th} bit positions are 1 will have one pair (the one making the transition from 0 to 1) plus the blocking nodes with a one in the k bit position plus one blocking node for each of the remaining pairs (the endpoints of such pairs that have a one in the k bit position). Therefore the resulting problem in the $(n-1)$-cube has 1 pair and q' blocking nodes, where $q' \leq p+q-1$. This problem can be solved by using our constructive proof for Lemma 1. The other subproblem in which all the k^{th} bit positions are 0 will have the $p-1$ pairs that make transitions from 1 to 0, plus the blocking nodes that have a zero in the k^{th} bit plus one endpoint for each pair that makes the transition from 0 to 1. This resulting problem is in the $(n-1)$-cube, has $p-1$ pairs and q' blocking nodes, where $q' \leq q+1$. The solution for this problem is obtained inductively (Theorem 1).

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 $1 \leq i \leq p < j \leq p+q$, $d(X_i) = n$, $d(X_j) = 0$, $p \leq \lceil n/2 \rceil$ and $2p+q \leq n+1$.*

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 in the conditions of Lemma 1 since $2p+q \leq n+1$ which implies that $q \leq n-1$. So assume that $p \geq 2$. By applying the 1-bit step we reduce the problem into two subproblems. The first problem is on $n-1$ dimensions and it has $p' = 1$ and $q' \leq n-2p+1+p-1 = n-p \leq n-1$ which fall into the conditions of Lemma 1. The second subproblem is on $n-1$ dimensions and it has $p' = p-1$ and $q' \leq n-2p+1+1$. Therefore, $p+q \leq n-2p+1+1+p-1 \leq n-1$ which holds for $p \geq 2$; hence, the resulting problem falls into the induction hypothesis. The theorem follows by induction. []

We use (n,p,q) to represent all problem instances in an n-cube with p paths and q blocking nodes. Note that these equivalence classes do not correlate with the underlying decision problem of whether or not node disjoint shortest paths exist between the endpoints. For example note that some instances of $(4, 2, 3)$ do not have node disjoint paths where as others do. For example $\{\{0000_2, 1111_2\}, \{0011_2, 1100_2\}, \{0101_2\}, \{0110_2\}, \{1001_2\}\}$ has no solution, but $\{\{0000_2, 1111_2\}, \{0011_2, 1100_2\}, \{0001_2\}, \{0110_2\}, \{1001_2\}\}$ does.

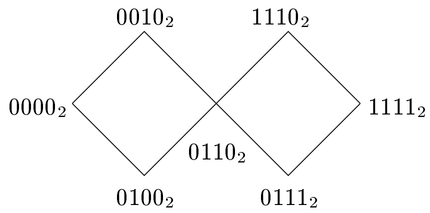
For the case when $n = 4$ it is impossible to strengthen our results because for $p = \lceil n/2 \rceil + 1 = 3$ all the problem instances do not have a solution, i.e., node disjoint shortest paths do not exist. One such instance is given in Example 1.

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

$$X = \{\{0000_2, 1111_2\}, \{1100_2, 0011_2\}, \{1010_2, 0101_2\}\}$$

has no 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. The only two nodes with exactly two bits equal to 1 that are not endpoints of the pairs in the 4-cube are 1001_2 and 0110_2. Therefore every path from 0000_2 to 1111_2 must go through one of these two nodes. Assume without loss of generality that the path goes through 0110_2 (the other case is identical after applying "exclusive or" with 1111_2 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_2 to 1111_2 in the following figure.



Nodes 0000_2, 0110_2 and 1111_2 must be part of any shortest path for pair X_1. We claim that the path for pair X_2 must visit node 1001_2. Suppose note. Suppose that the path for X_2 does not visit node 1001_2. Then the path for pair X_2 must either remain above or below the nodes on level 2 (nodes 0011_2, 0101_2, 1001_2, 0110_2, 1010_2 and 1100_2). Without loss of generality assume it remains above. Inspection of all the possibilities demonstrates that the path must include the node $0000_2 \in X_1$, but this is not allowed. Therefore every path for pair X_2 must use node 1001_2. 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 $k = 3$ (i.e. $0000_2 \oplus 2^3 = 1000_2$) as noted in the proof of Lemma 2. Applying the 1-bit step we end up with the problems $\{X_1 = \{000_2, 111_2\}, X_2 = \{100_2\}, X_3 = \{010_2\}\}$, and $\{X_1 = \{100_2, 011_2\}, X_2 = \{010_2, 101_2\}, X_3 = \{000_2\}\}$. The former problem can be solved, but not the latter one simply because there are not enough vertices in a 3-cube for two paths of length 3 and a blocking node.

As will be shown, the conditions of Theorem 1 can be improved 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 Theorem 1 does not cover this scenario.

Example 2 *Instance (n,p,q)=(5,4,0) with the pairs defined by*

$$X = \{\{00000_2, 11111_2\}, \{00001_2, 11110_2\}, \{00010_2, 11101_2\}, \{00100_2, 11011_2\}\}\ .$$

A set of disjoint paths for this problem are given by

$$\begin{array}{c}
00000_2 \leftrightarrow 01000_2 \leftrightarrow 01001_2 \leftrightarrow 01011_2 \leftrightarrow 01111_2 \leftrightarrow 11111_2 \\
00001_2 \leftrightarrow 00011_2 \leftrightarrow 00111_2 \leftrightarrow 00110_2 \leftrightarrow 01110_2 \leftrightarrow 11110_2 \\
00010_2 \leftrightarrow 01010_2 \leftrightarrow 11010_2 \leftrightarrow 11000_2 \leftrightarrow 11001_2 \leftrightarrow 11101_2 \\
00100_2 \leftrightarrow 00101_2 \leftrightarrow 10101_2 \leftrightarrow 10111_2 \leftrightarrow 10011_2 \leftrightarrow 11011_2
\end{array}$$

It is interesting to note that our proof technique (algorithmic approach) will not work for this problem instance. The reason is that the only transition that can be made is on bit $k = 0$ or $k = 1$. Since both cases are identical, under interchange of dimensions, we only discuss the case when the transition is on bit $k = 0$.
When a transition on bit $k = 0$ is made according to our 1-bit step, we advance as follows

$$\begin{array}{c}
11111_2 \rightarrow 01111_2 \\
00001_2 \rightarrow 10001_2 \\
00010_2 \rightarrow 10010_2 \\
00100_2 \rightarrow 10100_2
\end{array}$$

In the resulting problem the sub-cube in which bit 0 is always a one has the pairs

$$X = \{\{0001_2, 1110_2\}, \{0010_2, 1101_2\}, \{0100_2, 1011_2\}\}\ .$$

"Exclusive or"ing the input with 0001_2 yields the equivalent problem

$$X = \{\{0000_2, 1111_2\}, \{1100_2, 0011_2\}, \{1010_2, 0101_2\}\}.$$

This is just Example 1 and we know it does not have a solution. As in general there may be problem instances that are not amenable to the 1-bit change approach 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. The next section incrementally improves the sufficiency criteria by exploiting an extended version of Lemma 2.

3.2 The Balanced Approach: $p + q \leq n - 1$

Using the result of Theorem 1 one may obtain a stronger result. Given input n and $X = \{X_1, X_2, \ldots, X_{p+q}\}$ we claim that node disjoint shortest paths exist when $p + q \leq n - 1$ (Theorem 2). An algorithm based on the proof of Theorem 2 is given in Subsection 3.3.

We use the *1-bit balanced transformation* for finding disjoint paths for p pairs and $q = 0$ blocking nodes in an n-cube. This approach is similar in nature to the 1-bit step 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 this case one needs to apply an additional transformation which we call *q-dependent transformation*.

As in the 1-bit step, we start by selecting a bit k which we know exists by Lemma 2 since $p < n$. The reduction uses $\lceil p/2 \rceil$ transitions of the p pairs from 1 to 0 along the k^{th} bit and $\lfloor p/2 \rfloor$ transitions from 0 to 1 for the remaining pairs. The subproblem in which all the k^{th} bit positions are 1 will have the $\lfloor p/2 \rfloor$ pairs that make transitions from 0 to 1, plus one endpoint 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 k^{th} bit positions are 0 will have the $\lceil p/2 \rceil$ pairs that make transitions from 1 to 0, plus one endpoint 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. So another transformation needs to be applied, which we call the *q-dependent transformation*. We also need to prove a stronger version of Lemma 2 which can be applied to the above problem instances (Lemma 3).

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

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

*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 conflicting transition. Further 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

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

is of cardinality $4p+q$. So lets consider now the case when $p+q = n$. Since $p < n$ then $q \geq 1$. 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 $s_\ell = t_\ell$ 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 $s_\ell = t_{\ell'}$. Therefore s_ℓ is unique and therefore there is one conflict for bit k. Therefore the cardinality of set $e(X)$ for bit k is either $4p+q-1$ or $4p+q$. The lemma then 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. []

For 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 $p+q \leq n$ and $p < n$. First lets 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 k^{th} bit position. Without loss of generality one may assume that $i \leq \lfloor q/2 \rfloor$. 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 $\lfloor q/2 \rfloor$.

As it turns out we may assume that $p \geq q$ whenever we apply the q-dependent transformation. The transformation makes i transitions of the p pairs from 1 to 0 along the k^{th} bit and $p-i$ transitions from 0 to 1 for all of the remaining pairs. Therefore, the subproblem in which all the k^{th} bit positions are 1 will have the $p-i$ pairs that make initial transitions from 0 to 1, plus the i blocking nodes with a one on the kth bit position plus one endpoint of the i pairs that make the transition from 1 to 0. Therefore the resulting problem is an instance of $(n, p-i, 2i)$. The other subproblem in which all the k^{th} bit positions are 0 will have the i pairs that make transitions 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 make the transition from 0 to 1. This resulting problem is an instance of $(n, i, p+q-2i)$. In the algorithm presented in Section 3.3 we make extensive use of this transformation in procedure Find_Paths_1.

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 one in the k-bit position. Therefore a transition from a 1 to a 0 along bit k in a pair will not have a conflict with a blocking node. Since all pairs make this type of transition, it then follows that our transformation rules are valid. When $i > 0$ one may always select pairs for a 0 to 1 transition on bit k that do not have a conflict since at most $p/2$ pairs make a 0 to 1 transition. Therefore the above q-dependent transformation is valid on all cases.

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 $1 \leq i \leq p < j \leq p+q$, $d(X_i) = n > 6$, $d(X_j) = 0$, $p+q \leq n-1$ 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 $(n', p', q') = (6, 3, 3)$. As one may complement the vertices in the problem by $2^7 - 1 = 111111_2$; i, the number of blocked nodes with a 1 in the k^{th} bit position is in the range $0 \leq i \leq \lfloor 3/2 \rfloor = 1$. 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. 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 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. As mentioned before, we may assume that the number of blocked nodes with a 1 in the k bit positions is $0 \leq$

Table 1: Resulting Problem Instances

	Transitions	
i	$0 \rightarrow 1$	$1 \rightarrow 0$
0	$(5, 3, 0)$	$(5, 0, 6)$
1	$(5, 2, 2)$	$(5, 1, 4)$

Table 2: Resulting Problem Instances

	Transitions	
i	$0 \rightarrow 1$	$1 \rightarrow 0$
0	$(6, 4, 0)$	$(6, 0, 7)$
1	$(6, 3, 2)$	$(6, 1, 5)$

$i \leq \lfloor q/2 \rfloor = \lfloor 3/2 \rfloor = 1$. Thus, applying the general q-dependent transformation yields the instances given in Table 2.
Every instance of $(6, 0, 7)$ has a solution because there are only blocked nodes. Every instance of $(6, 1, 5)$ falls within the conditions of Lemma 1 and therefore the shortest path exists for every pair. 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 a conflict-free transition on bit k exists. 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 need only 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 a transition exists. Let k be that bit position. By transitioning half of the pairs along the bit k from 0 to 1 and the remaining pairs from 1 to 0 the two resulting subproblems are instances of

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

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.

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 k^{th} bit position. If $i > \lfloor q'/2 \rfloor$ then complement ($\oplus$) the subproblem input with $2^n - 1 = \underbrace{11\ldots1}_{n'}$. Therefore $0 \le i \le \lfloor q'/2 \rfloor$.

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

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

and

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

The former instance (n'', p'', q'') is within the induction hypothesis when $p'' + q'' \le n'' - 1$. 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,

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

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

Instance (n''', p''', q''') falls within the induction hypothesis when $p''' + q''' = q' + p' - i = p - i \le n - 3$. This holds true when $i \ge 2$. So let us now consider the remaining cases, $i \in \{0, 1\}$. When $i = 1$, Lemma 1 shows 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 2 it is clear why it is easier to prove it using Theorem 1 and Lemmas 2 and 3. In the next subsection we present our algorithm whose proof of correctness is reliant upon Theorem 2.

3.3 The Algorithm

An algorithm based on a straightforward implementation of Theorem 2 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 2. However, the algorithm does not really need to produce paths for the dummy pairs.

To reduce the time complexity the algorithm mimics the conversion of blocking nodes into pairs. If there are no (original) pairs, then the algorithm is done; if there is only one pair, then to find the paths it applies Lemma 1; and if there is more than one pair, it applies the constructive proof of Theorem 2 to construct the paths. When the dimension is 7 or 8 then it just applies the constructive proof for the base case of Theorem 2 to find the paths. The algorithm uses the proof of Theorem 2 in a two stage process to partition the problem. Our algorithm is initially invoked with Find_Paths_0(n, X).

3.3.1 Algorithm

Find_Paths_0$(n, X = \{X_1, X_2, \ldots, X_p, X_{p+1}, \ldots, X_{p+q}\})$
{
assume that $d(X_1) = d(X_2) = \ldots = d(X_p) = n, \;\; d(X_{p+1}) = d(X_{p+2}) = \ldots = d(X_{p+q}) = 0,$
    all the X_i pairs are in the same n dimensional sub$-$cube, and $p + q \leq n - 1$
if $p = 0$ return;
if $p = 1$ construct path using the proof of Lemma 1 and return;
if $n \leq 8$ return the paths constructed by the base case in Theorem 3;
    $k \leftarrow$ Find_Bit$_2(X)$; // using the proof of Lemma 2 find an unobstructed bit
    $X' \leftarrow$ OneToZeroBalancedTransition(k, X); //Transformation is defined below
    $X'' \leftarrow$ ZeroToOneBalancedTransition(k, X); //Transformation is defined below
    Output_Paths(k, X);
    Find_Paths_1$(n - 1, X')$;
    Find_Paths_1$(n - 1, X'')$;
}

The ZeroToOneBalancedTransition(k, X) and OneToZeroBalancedTransition(k, X) return the problem partitioned via the balanced transformation. The bit k is conflict-free due to the proof given in Lemma 2. 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 transition from a 0 to 1 on bit k. Endpoints of the remaining $\lfloor p/2 \rfloor$ with a one in the k^{th} bit position are returned as blocking nodes. Further q_0 is the number of singleton nodes with a one in the k^{th} 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 2 is simplified by converting blocking nodes to paths, the algorithm keeps actual pairs distinct from blocking nodes.

Find_Paths_1$(n, X = \{X_1, X_2, \ldots, X_p, X_{p+1}, \ldots, X_{p+q}\})$
assume that $d(X_1) = d(X_2) = \ldots = d(X_p) = n, \;\; d(X_{p+1}) = d(X_{p+2}) = \ldots = d(X_{p+q}) = 0,$
    all the X_i pairs are in the same n dimensional sub$-$cube, and $p + q \leq n$
if $p = 0$ return;
if $p = 1$ construct path per Lemma 1, output path and return;
if $n \leq 8$ 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 k^th 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'');
}
```

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

Function CountBlockingNodesWithOne(k, X) returns the number of blocking nodes with a one in the k^{th} 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.

Due to the aforementioned complementation function OneToZero_qTransition(i, k, X) does not have a conflict. This procedure is given p pairs and it selects $p - i$ of them for transition from 1 to 0. The residual blocking nodes from pairs is $2i$. However in the event that $i = 0$ and there is a conflict with a blocking node as indicated in Lemma 3 then an additional path is added to its output and blocking nodes with a zero in the k^{th} bit position are selected.

The function ZeroToOne_qTransition(i, k, X) returns i pairs which are capable of transitioning from 0 to 1 on bit k. If there is a conflict, as in Lemma 3, then when $i = 0$ one removes that pair from the output. If however $i > 0$ one simply selects i other paths for transition from 0 to 1. The number of residual blocking nodes with a one in the k^{th} bit position are selected.

3.4 Time Complexity

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 $1 \le i \le p < j \le p + q$, $d(X_i) = n > 6$, $d(X_j) = 0$, $p + q \le n - 1$ then algorithm* Finds_Paths_0 *constructs node disjoint shortest for X and can be implemented to take $O(m^{2.5})$ time, where m is the input length.*

Proof: The proof of correctness follows directly from the constructive proof of Theorem 2 and the fact that our algorithm just mimics the conversion of blocking nodes into pairs.

Both Find_Bit$_2$ and Find_Bit$_3$ are implemented using a binary trie for set insertion. The former call 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. For the n bits each insert has length n bits long and the number of such operations is $4p+q$. Therefore this operation will take at most $O(n^2(4p+q))$ 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 limited significantly at each level. The condition of Theorem 2 requires that $p<n$. At each level there are at most p invocations that have a at least one pair. Thus $2p$ problems are active at each of the n levels. Therefore the total number of calls is $O(2pn)$.

Hence, the algorithm runs in $O(2pn(n^2(4p+q))) = O(pn^3(4p+q)) = O(n^5)$. Since the input length is $m = O(n^2)$ the overall time complexity is $O(m^{2.5})$. []

3.5 Future Work

Our results are best possible in the sense that there are instances of (n,p,q) with $p+q=n$ that do not have node disjoint shortest paths. 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. As mentioned before determining whether or not node disjoint shortest paths exist in an n-cube is an NP-complete problem [5]. Therefore, necessary and sufficient conditions for the existence of node disjoint shortest paths are most likely not polynomial time computable.

References

[1] A. Aggarwal, J. Kleinberg, and D. P. Williamson. Node-Disjoint Paths on the Mesh and a New Trade-Off in VLSI Layout. *Siam J. Comput.*, 29(4):1321–1333, 2000.

[2] J. Ammon. Hypercube Connectivity within the ccNUMA Architecture. Silicon Graphics, May 1998.

[3] G. E. Andrews. *The Theory of Partitions.* Addison Wesley Publishing Company, 1976.

[4] S. Gao, B. Novick, and K. Qiu. From Hall's Matching Theorem to Optimal Routing on Hypercubes. *Journal of Combinatorial Theory, Series B*, 74(2):291–301, Nov. 1998.

[5] T. F. Gonzalez and F. D. Serena. n-Cube Network: Time Complexity of Node Disjoint Paths for k-Pairs of Vertices. Technical Report TRCS-16.01, University of California at Santa Barbara, September 2001.

[6] T. F. Gonzalez and 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 (PDCS'01)*, pages 278–282. IASTED, 2001.

[7] Q.-P. Gu and S. Peng. k-Pairwise Cluster Fault Tolerant Routing in Hypercubes. *IEEE Transactions on Computers*, 46(9), September 1997.

[8] Q.-P. Gu and S. Peng. Node-to-set and set-to-set cluster fault tolerant routing in hypercubes. *Parallel Computing*, 24:1245–1261, 1998.

[9] R. Karp. On the Computational Complexity of Combinatorial Problems. *Networks*, 5:45–68, 1975.

[10] S. Latifi, H. Ko, and P. K. Srimani. Node-to-Set Vertex Disjoint Paths in Hypercube Networks. *Computer Science Technical Report, Colorado State University*, CS-98-107, 1998.

[11] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* John Wiley & Sons, 1990.

[12] S. Madhavapeddy and I. H. Sudborough. A Topological Property of Hypercubes: Node Disjoint Paths. *Proc. Second IEEE Symp. Parallel and Distributed Processing*, pages 532–539, 1990.

[13] M. O. Rabin. Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance. *Journal of the Association of Computing Machinery*, 36(2):335–348, April 1989.

[14] C. Savage. A Survey of Combinatorial Gray Codes. *SIAM Review*, 39(4):605–629, Dec. 1997.

[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 Math. Journal*, 1968.