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

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

May 29, 2002

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 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 input length, and the second one takes $O(m^3)$ but it solves more general problem instances.

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

1 Introduction

The n-cube is a vital structure for parallel computing. Several systems with this communication architecture have been built. The SGI Origin 2000 is a 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 an efficient algorithm for a routing problem which is common to many applications.

The *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 $1 \le i \le p$, and $X_i = (a_i)$ for $p+1 \le i \le p+q$, find node disjoint shortest paths in the n-cube for all the pairs X_i that do not include blocking nodes and such that 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. Every node in the n-cube is represented by an n-bit string of bits 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 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. 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 is the shortest path in the graph between the two nodes independent from any other blocking nodes or endpoints of pairs $s_{i'}$ and $t_{i'}$ for $i' \neq i$. The nodes a_i (or blocking nodes or faulty processors) may also be included as part of the input. We show that when the distance between the endpoints of each pair X_i is extreme (i.e., $d(X_i) = n$ for $1 \le i \le p$), $p + q \le n - 1$ and $n > 6$ a solution always exists. Furthermore, we present an $O(m^3)$ time search algorithm which finds one such solution, where m is the input length.

Both the decision problems and the search problems are important for analysis. In the undirected n-cube there are yes and no instances of the k-pairwise node disjoint shortest paths decision problem. In the 2-cube $X = \{\{00, 11\}, \{01\}, \{10\}\}$ has no solution because any path between 00 and 11 must go through 01 and 10. On the other hand, $Y = \{\{00, 11\}, \{01\}\}$ does have such a solution with pair $\{00, 11\}$ yielding route $00 \leftrightarrow 10 \leftrightarrow 11$. The algorithms presented herein are search algorithms in the sense that they construct for yes instances of the decision problem a set of node disjoint paths. For problem instance Y given above a successful routing is $00 \leftrightarrow 10 \leftrightarrow 11$. Note that while problem instance $Z = \{\{000, 011\}, \{001\}, \{010\}\}$ has no shortest path solution, it does have a routing with any 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 $\{010\}$.

Node disjoint paths have been studied extensively. Karp [8] showed that determining whether or not there exist node disjoint paths for a set of pair of vertices in a graph is an NP-complete problem. Shiloach [14] presented a polynomial time algorithm to construct node disjoint paths in a graph for two pairs of vertices and Watkin [15] 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 [11] developed an $O(n^3 \log n)$ time algorithm to find disjoint paths for k pairs in an n-cube when $k > 2$ and $n \ge 4$. Each of the paths is of length at most $2n$. Then 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 $k \le \lceil n/2 \rceil$. In VLSI routing the problem finding disjoint paths in 1 and 2 dimensions has been extensively studied [1, 10]. 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 (i.e. for each pair X_i the length of the path must be $d(X_i)$). Rabin proposed an algorithmic strategy which addresses fault-tolerance, security and load balancing [12] . 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 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 each 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 pairs** 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 [7] also presents algorithms for the node-to-set node disjoint paths problem. Gao, Novick and Qiu [3] presented an algorithm for finding node-to-set node disjoint *shortest* paths. Gao, Novick and Qiu's paper [3] exploits 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 paths. Namely, as shown in [3], 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. 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 also address this issue for the n-cube, [3, 9]. The node-to-set problem reduces to our problem after finding the first link for each path and of course iterating over destination nodes. Finding k disjoint paths between two nodes in the hypercube has also been studied [6], but for brevity we do not discuss in detail these results.

Gonzalez and Serena [4] have shown 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. Since the problem of finding node disjoint shortest paths is NP-hard, we propose an algorithm for a restricted version of this problem.

The problem of finding node disjoint shortest paths in a mesh when one restricts all paths to a length of at most three is polynomially solvable [5]. It is NP-complete even when each of the pairs of vertices are at a sufficiently large constant distance. Gonzalez and Serena [5] have other very interesting results for the problem of finding edge disjoint shortest paths. For brevity we do not discuss this work in more detail.

In the next subsection we discuss the topology of the n-cube and then we define a restricted version of the problem and present algorithms for its solution.

1.1 Topology of an n-Cube

An n-cube, or equivalently an n dimensional hypercube, 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]

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

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 [13]. 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 [2]. Of course some edges are removed.

Since shortest paths in the n-cube are the primary focus of 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,

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

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

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

where $d(s, t)$ is the number of edges in a shortest path between nodes 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 [3] where ordered sets are used to denote the transition from a node to the next in a shortest path. For example the path

$$000 \to 010 \to 011 \to 111$$

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

1.2 The Extreme Search Subproblem

As mentioned before the node disjoint shortest path problem in the n-cube is an NP-complete problem. Therefore, it is unlikely that there exists a polynomial time algorithm for its solution.

[1] Hereafter binary numbers will appear without the subscript "$_2$".

In this subsection we present polynomial time algorithms for a restricted subproblem which we call the *extreme* version. The extreme version of the *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 $X_i = \{s_i, t_i\}$, $d(X_i) = n$, for $1 \le i \le p$, and $X_i = \{a_i\}$ for $p+1 \le i \le p+q$, find node disjoint shortest paths in the n-cube for all the pairs X_i that do not include blocking nodes and such that no two such paths have a node in common. Remember that the path for each pair X_i must have length equal to $d(X_i)$. For this version of the problem 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*. Our definitions imply that all the endpoints and blocking nodes are different, i.e., the cardinality of the set $e(X)$ is $2p+q$, where

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

For $1 \le i \le p$ each X_i is called a *pair* and the two nodes in it are called *endpoints* of X_i. The remaining X_i's are called blocking nodes.

Our first algorithm, for the case when $p \le \lceil n/2 \rceil$ and $2p + q \le n + 1$, constructs the set of node disjoint shortest paths by finding for each pair X_i one link incident to an endpoint of X_i which will be part of the final path and subsequently reduce 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 Section 2 and the algorithm, whose time complexity is $O((p^2 + q)n^2)$, is given in Subsection 2.1. Latifi, Ko and Srimani [9] 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 second algorithm, for the case when $p + q \le n - 1$, is based on the proof of Theorem 3 in Section 3. This algorithm finds a link for each pair and reduces the problem, using a balanced transformation, to finding a set of node disjoint paths in an $(n-1)$-cube. Since the two resulting subproblems do not fall within the induction hypothesis, 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 yield an algorithm which generates paths in a time efficient manner. The algorithm, whose time complexity is $O((p^2 + pq)n^3)$, is given in Subsection 3.1.

The sub-cube 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 when $s = t$ then the subcube 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.

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

In this section we prove that when $p \leq \lceil n/2 \rceil$ and $2p+q \leq n+1$ node disjoint shortest paths exist for the extreme version of our problem. As shown in subsection 2.1, our constructive proofs can be easily implemented to take $O((p^2+pq)n^2)$ time. The shortest paths we construct have a fixed structure. The structure is characterized by a permutation of the dimensions $\overline{\pi} = (\pi_1\pi_2\ldots\pi_n) \in S_n$.[2] Each of the paths is such that the first or last link in the path is dimension π_1 and exactly one such path transitions from a one to a zero. The other $p-1$ paths transition from a zero to a one. After deleting such a link from each path the property holds for the $p-1$ paths in subsequent dimensions using the remaining part of permutation $\overline{\pi}$. Our algorithmic approach, based on the proof of Theorem 1, is to construct the set of node disjoint shortest paths as follows.

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, $1 \leq i \leq p$, 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 with 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 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 sub-cube $K(010, 111)$ with blocking node 011 and finishing up the path for X_2 in the sub-cube $K(001, 100)$ 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 transition introduced on bit k and we get the node disjoint shortest paths in the 3-cube.

[2] S_n is the permutation of n dimensions.

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 $q \leq n-1$ blocking nodes in an n-cube for all $n \geq 1$, 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 $n \geq 1$. 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 $n > 1$. 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 $2^0, 2^1, \ldots, 2^{n-1}$. 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 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. []

The constructive proof of Lemma 1 can be used to develop an $O(n^2)$ time algorithm to construct the path for the single pair in the presence of at most $n-1$ blocking nodes. We assume that $X_1 = (000...0, 111...1)$. The algorithm builds a vector $(v[1, n])$ with the number of blocking nodes at a distance i, for $1 \leq i < n$, from 000...0 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...0. If the entry $v[1]$ is nonzero then the path path for pair X_1 takes 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 a 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 2^j. The resulting problem is solved inductively without having to generate the data structure again. The algorithm can be shown to take $\sum_{j=1}^{n-1} v[j] + 1$, which is $O(n^2)$.

Our algorithmic approach for the simple problem is to find a bit k on which to transition. But our selection for such a bit is actually more restricted than we have indicated. We find a bit k such that if every endpoint of a pair transitions on bit k then no endpoint of a pair or blocking node will be overlapped on the transition. Note that in our algorithm we only 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 be 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 $1 \le i \le p$, 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(s) 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 the paths for all pairs.

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 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 tuples 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 between the endpoints of pair X_{h_i}, in the

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.

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 $p<n$. It is seen that the lemma holds for the criteria that $p+q<n$. Since when $q \geq 1$ 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. []

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 1 to 0 along the k^{th} bit and $p-1$ transitions from 0 to 1 for the remaining pairs. Therefore, the subproblem in which all the k^{th} 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 k^{th} 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 $q' \leq p-1+q$. 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 1 will have the $p-1$ pairs that make transitions from 0 to 1, plus the blocking nodes that have a one in the k^{th} 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 $p-1$ 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 $2p+q \leq n+1$ 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 $q' \leq n+1-2p+p-1 = n-p < n-1$ which falls into the conditions of Lemma 1. The second subproblem is on $n-1$ dimensions and it has $p'=p-1$ and $q' \leq n+1-2p+1$. Therefore, $p'+q' \leq n+1-2p+1+p-1 \leq n-1$ 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 this collection of problem instances does not correlate with the underlying decision problem of whether or not node disjoint shortest paths exist between the endpoints.

There are problem instances of $(4,2,3)$ that do not have node disjoint paths where as 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. But $\{\{0000,1111\},\ \{0011,1100\},\ \{0001\},\ \{0110\},\ \{1001\}\}$ does ($0000 \leftrightarrow 0100 \leftrightarrow 0101 \leftrightarrow 0111 \leftrightarrow 1111$, and $0011 \leftrightarrow 0010 \leftrightarrow 1010 \leftrightarrow 1000 \leftrightarrow 1100$).

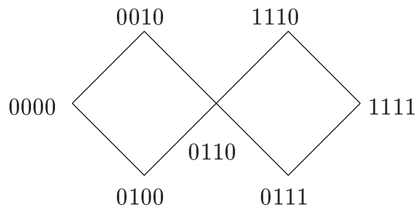
For the case when $n = 4$ it is impossible to strengthen our results because for $p = \lceil n/2 \rceil + 1 = 3$ all the problem instances do not have a solution, i.e., node disjoint shortest paths do not exist. One such instance is given in Example 1.

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

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

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 the following figure.



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 note. 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 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.

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 are given by

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

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 the only transition that can be made is on bit $k = 3$ or $k = 4$. Since both cases are identical, under interchange of dimensions, we only discuss the case when the transition is on bit $k = 4$.
When a transition on bit $k = 4$ is made according to our 1-bit step, we advance as follows

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

In the resulting problem the sub-cube 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. 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.

The reason why our algorithmic approach does not work in this case even though there is a solution, is that our approach finds a solution with the structure defined at the beginning of this section, but such a solution does not exist for the instance given in Example 2.

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 Theorem 1 to partition the problem. The program is called initially with Find_Paths(n, X).

Find_Paths$(n, X = \{X_1, X_2, \ldots, X_p, X_{p+1}, \ldots, X_{p+q}\})$
{
assume that $n = d(X_1) = d(X_2) = \ldots = d(X_p)$, $0 = d(X_{p+1}) = d(X_{p+2}) = \ldots = d(X_{p+q})$,
all X_i are in the same n dimensional sub$-$cube, $p \leq \lceil n/2 \rceil$ and $2p + q \leq n + 1$
if $p = 0$ return;
if $p = 1$ construct path using the proof of Lemma 1, output and return;
if $n \leq 2$ return appropriate base case;
$k \leftarrow$ Find_Bit(X); // using the proof of Lemma 2 find an unobstructed bit.
$X' \leftarrow$ OneToZeroTransition(k, X);
$X'' \leftarrow$ 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 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 k^{th} bit position are selected. ZeroToOneTransition selects $p - 1$ paths for 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 k^{th} 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 $1 \leq i \leq p < j \leq p + q$, $d(X_i) = n$, $d(X_j) = 0$, $p \leq \lceil n/2 \rceil$, $2p + q \leq n + 1$ then algorithm* Finds_Paths *constructs node disjoint shortest paths for X and can be easily implemented to take $O((p^2 + q)n^2)$ 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 Lemma 2. Procedure Find_Bit is implemented by using a binary trie 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 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 the transitions 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 $O(pn^2(p+q))$. []

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

The problem given in Example 3 has a node disjoint shortest path solution. If we apply our algorithmic strategy, the only bit transitions 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 $k = 5$. Now 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. Therefore our algorithm in this section does not work here, but there is a solution to the problem as well as for all problem instances in $(n,p,q) = (7,6,0)$. In the next section we give an improved algorithm that works for $(n,p,q) = (7,6,0)$ as well as in more general cases.

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

$$\begin{aligned} X &= \{\{0000000, 1111111\}, \{0000001, 1111110\}, \\ &\quad \{0000010, 1111101\}, \{0000100, 1111011\}, \\ &\quad \{0001000, 1111011\}, \{0010000, 1101111\}\} \end{aligned}$$

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

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 $p + q \leq n - 1$ (Theorem 3). An algorithm based on the proof of Theorem 3 is given in Subsection 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 suppose now that the instance of (7,6,1) has a bit for transition. If we apply again the 1-bit approach we end up with instances of (6,1,7) ands (6,5,2). Both of these instances might not 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*. We apply this 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 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 is like 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.

For example if we start with an instance of $(8, 7, 0)$ we end up with instances of $(7, 4, 3)$ and $(7, 3, 4)$. Clearly, these two instances have the two undesirable properties. But in Lemma 3 we establish 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* to reduce our problems to ones that fall within the induction hypothesis. 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 in one transition from an endpoint in a pair to a blocking node.

The above situation arises when considering the following instance (8,7,0): $X = \{\{00001111, 11110000\}, \{01000111, 10111000\}, \{01100011, 10011100\}, \{01110001, 10001110\}, \{10110000, 01001111\}, \{10011000, 01100111\}, \{10001100, 01110011\}\}$. Applying the 1-bit balanced transformation we obtain instances of $(7, 4, 3)$ and $(7, 3, 4)$. The former instance is obtained by making 1 to 0 transitions along bit $k = 7$ in the first 4 pairs and 0 to 1 transitions in the remaining pairs. The resulting problem instance of $(7, 4, 3)$ is $X = \{\{0001111, 1110000\}, \{1000111, 0111000\}, \{1100011, 0011100\}, \{1110001, 0001110\}, \{1001111\}, \{1100111\}, \{1110011\} = \{\{s_1, t_1\}, \{s_2, t_2\}, \{s_3, t_3\}, \{s_4, t_4\}, \{x_1\}, \{x_2\}, \{x_3\}\}$. This instance does not have a conflict free bit k as the one which is identified by Lemma 2. There is not a conflict free bit.

This is clearly an example of a problem that does not have a 0 to 1 bit transition available; however it does have the transition from 1 to 0 on bit $k = 6$.

$$
\begin{aligned}
t_4 \oplus 2^0 = &\quad 0001110 \oplus 2^0 &= 0001111 = s_1 \\
s_4 \oplus 2^1 = &\quad 1110001 \oplus 2^1 &= 1110011 = x_7 \\
s_3 \oplus 2^2 = &\quad 1100011 \oplus 2^2 &= 1100111 = x_6 \\
s_2 \oplus 2^3 = &\quad 1000111 \oplus 2^3 &= 1001111 = x_5 \\
s_3 \oplus 2^4 = &\quad 1100011 \oplus 2^4 &= 1110011 = x_7
\end{aligned}
$$

$$s_2 \oplus 2^5 = \quad 1000111 \oplus 2^5 \quad = 1100111 = x_6$$
$$s_1 \oplus 2^6 = \quad 0001111 \oplus 2^6 \quad = 1001111 = x_5$$

Note that this problem instance illustrates that Lemma 3 is required for this problem instance. Further the example is applicable in the execute of the algorithm presented in Section 3.1.

The above example can be easily extended to higher dimensions, demonstrating that the bit k that satisfies the properties of Lemma 2 is not possible. Therefore in Lemma 3 we establish 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 in 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 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 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 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 we show later on we may assume that $p \geq q$ whenever we apply the q-dependent transformation. The transformation makes i transitions from an endpoint of p pairs from 1 to 0 along the k^{th} 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 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 k^{th} bit position plus one endpoint of the i pairs that make 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 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 $(n, i, p + q - 2i)$. Our algorithm Find_Paths_1 given in Section 3.1 makes extensive use of this transformation.

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 a 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 then follows that our transformation rules are valid. When $i > 0$ one may always select pairs for a 1 to 0 transition on bit k that do not have the conflict and such a pair exists because at most $p/2$ pairs make a 1 to 0 transition. 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 in the k^{th} bit transition 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 $1 \leq i \leq p < j \leq p+q$, $d(X_i) = n > 6$, $d(X_j) = 0$, $p + q \leq n - 1$ 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 $(n', p', q') = (6, 3, 3)$. Each of these subproblems satisfies the conditions of Lemma 2, therefore a bit k for the transition exists but it may have one conflict. As one may complement the vertices in the problem by $2^7 - 1 = 111111$; 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. Note that we can always perform these transitions 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.

Table 1: Resulting Problem Instances for $(6, 3, 3)$

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

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 transitions that may have a conflict. As mentioned before, we may assume that the number of blocked nodes with a 1 in the k bit positions is $0 \leq i \leq \lfloor q/2 \rfloor = \lfloor 3/2 \rfloor = 1$. Thus, applying the general q-dependent transformation yields the instances given in Table 2. Note that we can always perform these transitions while avoiding the conflict identified in Lemma 2.

Table 2: Resulting Problem Instances for $(7, 4, 3)$

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

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 a shortest path exists for the 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 when needed. 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 $p' \geq q'$ and therefore thus 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 k^{th} bit position. If $i > \lfloor q'/2 \rfloor$ then complement ($\oplus$) the subproblem input with $2^n - 1 = \underbrace{11\ldots1}_{n'}$. Thus $0 \leq i \leq \lfloor q'/2 \rfloor$ holds.

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

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

and

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

The reason why it is possible to apply these two transformations is that we can always select the pair with the conflict fall into the appropriate transition and avoid the conflict. When i is equal to zero all the transitions for the pairs are 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 into the appropriate transition.

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

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

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 $p''' + q''' = q' + p' - i = p - i \leq n - 3$. This holds true when $i \geq 2$. 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 3 it is clear why it is easier to prove it using Theorem 1 and Lemmas 1, 2 and 3. In the next subsection we present our algorithm whose proof of correctness is reliant 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 the algorithm mimics 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 paths it applies Lemma 1; and if there is more than one pair, it applies the constructive proof of Theorem 3 to construct 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$(n, X = \{X_1, X_2, \ldots, X_p, X_{p+1}, \ldots, X_{p+q}\})$
{ 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 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 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. We use q_0 to denote 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 3 is simplified by converting blocking nodes to paths, the algorithm keeps actual pairs distinct from blocking nodes.

Find_Paths_1$(n, X = \{X_1, X_2, \ldots, X_p, X_{p+1}, \ldots, X_{p+q}\})$
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 { // $p + q = n$
  $k \leftarrow$ 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$(k, X) > \lfloor q/2 \rfloor$) then $X =$ ComplementProblem(n, X);
  //the problem must be complemented appropriately upon output.
  $i \leftarrow$ CountBlockingNodesWithOne(k, X);
  $X' \leftarrow$ OneToZero_qTransition(i, k, X); //Transformation is defined below
  $X'' \leftarrow$ 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 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.

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 and it selects $p - i$ of them for 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 is not a problem because all the blocking nodes have their k bit equal to 0 but all transitions are from a 0 to a 1. When $i > 0$ one can always sect the pair with the conflicting transition into the appropriate transition that will avoid the conflict.

Similarly the function OneToZero_qTransition(i, k, X) returns i pairs which are capable of transitioning 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 transition from 0 to 1. The number of residual blocking nodes with a one in the k^{th} bit position placed into the input to this function. When $i = 0$ this type of transition is not used.

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 $1 \leq i \leq p < j \leq p + q$, $d(X_i) = n > 6$, $d(X_j) = 0$, $p + q \leq n - 1$ then algorithm* Finds_Paths_0 *constructs node disjoint shortest paths for X and can be implemented to take $O((p^2 + q)n^3)$ 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. 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 $O(n^2(p + q))$ 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 $O(pn(n^2(p + q))) = O((p^2 + pq)n^3)$.
[]

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

4 Future Work

Our results are interesting since there is always a node disjoint shortest path solution. Note 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. One caveat however, determining whether or not node disjoint shortest paths exist is an NP-complete problem in an n-cube [4, 5]. 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 to be NP-complete.

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

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

[4] T. F. Gonzalez and F. D. Serena. n-Cube Network: Complexity of Node Disjoint Paths for k-Pairs of Vertices. Technical Report TRCS-2001-16, University of California at Santa Barbara, September 2001.

[5] T. F. Gonzalez and 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, April 2002.

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

[7] 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.

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

[9] 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.

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

[11] 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.

[12] 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.

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

[14] Y. Shiloach. Two Paths Problem is Polynomial. Technical Report TR-CS-78-654, Stanford University, 1978.

[15] M. Watkin. Graph is (2k-1)-connected is a necessary condition to admit k paths. *Duke Math. Journal*, 1968.