Parallel Processing Letters
© World Scientific Publishing Company

ON THE COMPUTATIONAL COMPLEXITY OF ROUTING IN FAULTY *K*-ARY *N*-CUBES AND HYPERCUBES

Iain A. Stewart
School of Engineering and Computing Sciences, Durham University
Science Labs, South Road, Durham DH1 3LE, U.K.

Received (received date)
Revised (revised date)
Communicated by Ke Qiu

ABSTRACT

We equate a routing algorithm in a (faulty) interconnection network whose underlying graph is a k-ary n-cube or a hypercube, that attempts to route a packet from a fixed source node to a fixed destination node, with the sub-digraph of (healthy) links potentially usable by this routing algorithm as it attempts to route the packet. This gives rise to a naturally defined problem, parameterized by this routing algorithm, relating to whether a packet can be routed from a given source node to a given destination node in one of our interconnection networks in which there are (possibly exponentially many) faulty links. We show that there exist such problems that are **PSPACE**-complete (all are solvable in **PSPACE**) but that there are (existing and popular) routing algorithms for which the computational complexity of the corresponding problem is significantly easier (yet still computationally intractable).

Keywords: routing; computational complexity; k-ary n-cubes; hypercubes

1. Introduction

Routing is the problem of devising algorithms to route a packet from a source node to a destination node in a given interconnection network, and is fundamental to parallel and distributed computing. Routing algorithms are commonly evaluated according to such criteria as the length of the eventual path traversed, the resulting size of the header of a packet (in an implementation of the routing algorithm), the adaptivity of the routing algorithm or whether the routing algorithm is free of deadlocks or livelocks (see, for example, [4] for a scheme for the classification of routing algorithms).

Another important requirement of a routing algorithm is that it can tolerate (a limited number of) faults in an interconnection network, and this requirement will constitute our focus in this paper. If we fix a source node and a destination node in some interconnection network in which there might be faults (either faulty nodes or faulty links), corresponding to many routing algorithms is a sub-digraph R of the interconnection network consisting of the links that might be used by the routing algorithm in order to route a packet from the source to the destination (throughout, our interconnection networks are regarded as digraphs resulting from a k-ary n-cube Q_n^k or a hypercube Q_n in which every edge has been replaced by a pair of directed edges of opposite orientations). Moreover, the links of this sub-digraph R can usually be specified by an algorithm that is applied at each node of the interconnection network in a distributed fashion (many routing algorithms are such that the links traversed from the current node are determined by the current node, the destination node and possibly some other parameters). In this paper, we equate a routing algorithm with its corresponding digraph R and we consider the computational complexity of the problem of deciding whether there is a path in R from the source node to the destination node (we provide more explanation in the next section).

One might think that because we are working in Q_n^k and Q_n, which have k^n and 2^n nodes, respectively, any complexity-theoretic analysis will necessarily involve resources that are exponential in n. However, this is not so, for we do not specify our interconnection networks explicitly, as adjacency matrices, but implicitly via the parameters k and n (or just n in the case of hypercubes). Also, as we have hinted above, our sub-digraph R is not explicitly specified but is given via an algorithm that is to be applied at any node so as to yield the non-faulty links emanating from that node within R. In this note, we prove that in this context there are such routing problems for interconnection networks resulting from k-ary n-cubes and hypercubes that are **PSPACE**-complete but that this complexity is reduced (to **NP**-complete) for particular routing algorithms (every such routing problem can be solved in **PSPACE**). The particular routing algorithms we choose are dimension-order routing and negative-first routing. Our results provide the first complexity-theoretic analysis of routing in the presence of (possibly an exponential number of) faults in specific interconnection networks prevalent in parallel computing and demonstrate how even the most straightforward of routing algorithms has a number of nuances and can be computationally intractable.

2. Connectivity in the presence of faults

As mentioned above, we work exclusively with k-ary n-cubes and hypercubes. A *k-ary n-cube* Q_n^k has a vertex set of $V(Q_n^k) = \{0, 1, \ldots, k-1\}^n$ and there is an edge $((u_1, u_2, \ldots, u_n), (v_1, v_2, \ldots, v_n))$ if, and only if, $|u_i - v_i| = 1 \pmod{k}$, for some $i \in \{1, 2, \ldots, n\}$, with $u_j = v_j$, for all $j \in \{1, 2, \ldots, n\} \setminus \{i\}$; such an edge is termed as lying in *dimension i* (throughout, arithmetic on the components of vertices is modulo k). A *hypercube* of dimension n has a vertex set of $V(Q_n) = \{0, 1\}^n$ and there is an edge $((u_1, u_2, \ldots, u_n), (v_1, v_2, \ldots, v_n))$ if, and only if, $|u_i - v_i| = 1$, for some $i \in \{1, 2, \ldots, n\}$, with $u_j = v_j$, for all $j \in \{1, 2, \ldots, n\} \setminus \{i\}$; again, such an edge is termed as lying in *dimension i*. Our interconnection networks, which we also refer to as Q_n^k and Q_n, are obtained by replacing every edge in the undirected graphs Q_n^k and Q_n with two directed edges of opposite orientations. Henceforth, all references to Q_n^k and Q_n are as interconnection networks and so we prefer the terminology 'nodes' and 'links' as opposed to 'vertices' and 'edges'.

2.1. *Implicitly describing interconnection networks*

Consider the k-ary n-cube Q_n^k or the hypercube Q_n. Everything we say below applies equally to hypercubes so we only focus here on k-ary n-cubes. Trivially, Q_n^k is connected. However, if there are some faulty nodes or links within Q_n^k then this may not be the case (later, we will work exclusively with faulty links but for the moment we leave it open as to whether we might have faulty nodes or faulty links). We are interested in the computational complexity of certain connectivity problems in faulty k-ary n-cubes. As regards connectivity problems in arbitrary graphs, ordinarily such a complexity analysis would be parameterized by the number of nodes in the graph and we would be looking for efficient algorithms or hardness results where the size of an instance of any of our problems is determined by the number of nodes in the input graph. However, our input graphs are always k-ary n-cubes and so we do not need to explicitly present an input instance as, for example, a $k^n \times k^n$ adjacency matrix; all we need supply are the parameters k and n. Consequently, our analysis will focus upon computational resources that are polynomial in the parameters k and n rather than the parameter k^n (in fact, k will soon be regarded as a constant). The main advantage of adopting such a position is that, as we shall see, doing so allows us to consider sub-exponential (in terms of n) resources in, for example, distributed environments where a routing algorithm does not have global knowledge relating to the faults of a faulty k-ary n-cube but discovers the faults as it progresses.

2.2. *Implicitly describing faults*

Having explained how we implicitly describe our k-ary n-cubes, we now consider mechanisms for detailing the faults within our k-ary n-cubes. Given that our k-ary n-cubes will be represented implicitly, so that the size of any instance of one of our problems is bounded by some polynomial in k and n, it would appear that we are limited to prescribing a polynomial number (in k and n) of faults within any k-ary n-cube. However, just as we have presented k-ary n-cubes implicitly, so we do with the faults within any k-ary n-cube. In particular, our connectivity problems are parameterized by algorithms which specify the faults at (that is, local to) any given node of a k-ary n-cube (such faults might be faulty links emanating from a node or faulty neighbours of that node). Not only does this approach allow us to specify a potentially exponential number (in n) of faults but it also allows us to focus on particular distributed routing algorithms.

Example 2.1. Consider dimension-order routing in Q_n^k (also known as e-cube routing) where the dimensions are ordered via some fixed ordering, such as $1, 2, \ldots, n$, and where a packet is routed across each dimension in turn (in dimension order) so that it moves one hop closer to the destination along a shortest path [3] (for the sake of completeness, whenever there is a choice of shortest paths our routing algorithm always increments, rather than decrements, a component). We can formulate this routing algorithm in our framework by defining an algorithm α which for any node (which we call the current node): compares its bits with those of the destination node (one by one and in dimension order); finds the first dimension d where they differ; and marks all links from the current node as faulty except for the link in dimension d to be followed via the dimension-order routing algorithm. Any path from a source node to a destination node undertaken by dimension-order routing in Q_n^k corresponds to a path from the source node to the destination node in Q_n^k where the algorithm α is applied at each node to detail the faulty links emanating from that node, and *vice versa*. If we wished to relax dimension-order routing so that whenever there is a choice of shortest paths at some node, both paths are available (and so k is necessarily even), then we could simply amend our algorithm α so that both such exit links from the node are made healthy, with the rest designated as faulty. Alternatively, we might amend α to impose additional faults within Q_n^k and we might be interested as to whether it is still possible for dimension-order routing to route a packet from a given source node to a given destination node even in the presence of these faults. Dimension-order routing results in a particularly simple scenario as the sub-digraph induced by the links allowable via dimension-order routing is such that every node has out-degree at most 1, and so given that a path from the source to the destination (under dimension-order routing in a non-faulty k-ary n-cube) has length at most $n\lfloor\frac{k}{2}\rfloor$, we can solve our connectivity problem in polynomial-time (assuming that we can ascertain the healthy link emanating from a node, if there is one, in polynomial-time). Indeed, it can easily be seen that our relaxed dimension-order routing algorithm, above, results in a polynomial-time connectivity problem.

Consequently, our connectivity problems will be relative to a specific algorithm α that can be considered as describing both the routing algorithm and the faults in some k-ary n-cube. Let us look at some properties we require of the algorithm α and some reasoning as to why.

The algorithm α should be such that given a node, the faults local to that node, as described by α, can be ascertained ‘quickly’. This restriction is made because it is the search within a faulty k-ary n-cube upon which we wish to focus rather than the complexity of determining the faults at some node. This is in keeping with a scenario where a distributed routing algorithm, for example, can immediately ascertain the faults local to a specific node.

Our algorithm α should be such that as well as the parameters k, n and the current node at which α is to be applied, there are additional parameters which are input to α. If this were not the case then for any fixed k and n, there would be exactly one faulty k-ary n-cube (parameterized by α). Such a scenario lacks flexibility. Take dimension-order routing, for example. Here, the routing algorithm is parameterized not only by k, n and the current node but by the destination node and the order imposed upon the dimensions. Also, if we are to use our framework in order to specify faulty k-ary n-cubes within which connectivity problems need to be solved or distributed routing algorithms need to be tested then we require more flexible situations than the one where there is only 1 fault scenario for every k and n. Note that our framework allows us to use our algorithm α to specify (what amounts to) a routing algorithm and any (fixed) polynomial number (in n) of faults, which is a scenario much studied in interconnection networks (where the basic assumption is often that the number of faults occurring within an interconnection network will ordinarily be small, i.e., poly-logarithmic, in comparison to the number of nodes of the network).

2.3. *Our connectivity problems*

We are now in a position to define our core connectivity problem. Let α be some algorithm that is parameterized by k, n, the current node and additional parameters so that: the number of additional parameters is bounded by some polynomial in k and n; α runs in time bounded by some polynomial in k and n; and for any node of Q_n^k, α specifies the links emanating from the current node that are not faulty (on input k, n, the current node and fixed values for the additional parameters). Henceforth, we will call such an algorithm a *fault-describing* algorithm.

Definition 2.1. The problem st-Connectivity-in-k-ary-n-cubes(α) is defined as follows:

- an instance of size nk is an implicitly specified copy of Q_n^k together with a source node $\mathbf{u}$, a destination node $\mathbf{v}$ and some values W so that the parameters of α are given fixed values from $\mathbf{u}$, $\mathbf{v}$ and W;
- a yes-instance is an instance for which it is possible to route a packet from the source node $\mathbf{u}$ to the destination node $\mathbf{v}$ so as to avoid the faulty links as specified by the algorithm α (w.r.t. k, n, the current node, $\mathbf{u}$, $\mathbf{v}$ and W).

It is trivial to convince oneself that all problems st-Connectivity-in-k-ary-n-cubes(α) are solvable in **PSPACE** (simply iteratively guess a path one node at a time, checking that it is a legitimate one as the path is traversed, and use the fact that **PSPACE** = **NPSPACE**).

The problems in Definition 2.1 are in their most general form. We can fix the value of k, for any $k \geq 3$, and obtain the problems st-Conn-$Q_n^k(\alpha)$, and it is these problems that we consider in what follows. We emphasis that in the problems st-Conn-$Q_n^k(\alpha)$, k is fixed (at, for example, $k = 8$) and all instances are relative to Q_n^k (for example, to Q_n^8). We also define the problems st-Conn-$Q_n(\alpha)$, obtained by setting $k = 2$ and thus working with hypercubes as opposed to k-ary n-cubes.

Example 2.2. As an illustration of a specific connectivity problem, formulated within our framework, let α be the dimension-order routing algorithm for Q_n^k where the additional parameters supply the destination node, an ordering of the dimensions and n specific faulty links within Q_n^k (more precisely, α details the non-faulty links emanating from the current node under dimension-order routing and taking into account the additional n faulty links). The problem ST-ROUTCONN-$Q_n^k(\alpha)$ is the problem of deciding whether it is possible to route a packet from a given source to a given destination within Q_n^k via dimension-order routing, according to the given order of dimensions and so as to avoid using any of the additional n faulty links. Of course, as remarked earlier, this particular problem is straightforward to solve in polynomial-time.

It is worthwhile mentioning some other aspects of our fault framework. As we have stated, our framework allows us to specify (what amounts to) a routing algorithm together with any list of a (fixed) polynomial number of faulty links. Whilst we can also specify collections of exponentially many (in n) faults in Q_n^k, we clearly cannot specify every such collection. Nevertheless, there are numerous interesting patterns of exponentially many faults that we can specify. For example, suppose that we were interested in the performance of some particular routing algorithm when a requirement is as follows: the fault configuration is such that the faults are faulty nodes so that any two such faulty nodes are relatively distant from one another. We could use α to specify not only the routing algorithm but also that all links incident with any node of

$$\{(u_1, u_2, \ldots, u_n) \in V(Q_n^k) : \text{each } u_i \text{ is equal to 0 or } \lfloor\frac{k}{2}\rfloor\}$$

are faulty (this is equivalent to saying that the nodes of this set are faulty). This pattern of faults is exponential (in n) in size and is such that that no two 'faulty nodes' are less than a distance $\lfloor\frac{k}{2}\rfloor$ apart.

3. Some connectivity results

We begin by classifying the complexity of the problem ST-CONN-$Q_n^k(\alpha)$ when α is an arbitrary fault-describing algorithm (we already have an upper bound of **PSPACE** for this problem but no lower bounds). In order to do this, we need to define the problem k-COLOUR-PATH, where $k \geq 2$.

Let G be an undirected graph on n vertices. The graph $\mathcal{C}_k(G)$, known as the *k-colour graph*, has vertex set consisting of all proper k-colourings of G, and is such that there is an edge joining two k-colourings if one k-colouring differs from the other with respect to the colour of exactly one vertex (note that the k-colour graph of G is empty if G can not be properly k-coloured). We identify (not necessarily proper) k-colourings of G and elements $\mathbf{x} \in \{0, 1, \ldots, k-1\}^n$ via: vertex i is coloured x_i, for $i = 1, 2, \ldots, n$. The decision problem k-COLOUR-PATH has as an instance a graph G and two distinct, proper k-colourings, and this instance is a yes-instance if there is a path in $\mathcal{C}_k(G)$ joining the two k-colourings. It was proven in [1] that if $k \geq 4$ then k-COLOUR-PATH is **PSPACE**-complete, but that if $k = 3$ then k-COLOUR-PATH can be solved in polynomial-time.

Theorem 1. For each $k \geq 3$, there exists a fault-describing algorithm α so that the problem ST-CONN-$Q_n^k(\alpha)$ is **PSPACE**-complete.

Proof. We begin by assuming that $k \geq 4$. Let G be an undirected graph on n vertices and let $\mathbf{u}$ and $\mathbf{v}$ be 2 distinct proper k-colourings of G. For any $\mathbf{x} \in \{0, 1, \ldots, k-1\}^n$, let $G(\mathbf{x})$ denote the graph G coloured so that vertex i is given the colour x_i, for all $i = 1, 2, \ldots, n$. For every $\mathbf{x} \in \{0, 1, \ldots, k-1\}^n$, make the following links of Q_{2n}^k healthy, for $i = 1, 2, \ldots, n$, with all other links faulty:

$$\begin{aligned}
&(x_1, 0, \ldots, x_{i-1}, 0, x_i, 0, x_{i+1}, 0, \ldots, x_n, 0) \rightarrow \\
&\quad (x_1, 0, \ldots, x_{i-1}, 0, x_i, 1, x_{i+1}, 0, \ldots, x_n, 0) \\
&\qquad \text{if } G(\mathbf{x}) \text{ is a proper colouring;} \\
&(x_1, 0, \ldots, x_{i-1}, 0, x_i, 1, x_{i+1}, 0, \ldots, x_n, 0) \rightarrow \\
&\quad (x_1, 0, \ldots, x_{i-1}, 0, x_i \pm 1, 1, x_{i+1}, 0, \ldots, x_n, 0); \\
&(x_1, 0, \ldots, x_{i-1}, 0, x_i, 1, x_{i+1}, 0, \ldots, x_n, 0) \rightarrow \\
&\quad (x_1, 0, \ldots, x_{i-1}, 0, x_i, 0, x_{i+1}, 0, \ldots, x_n, 0) \\
&\qquad \text{if } G(\mathbf{x}) \text{ is a proper colouring.}
\end{aligned}$$

It is trivial to verify that the resulting faulty k-ary $2n$-cube can be described by a fault-describing algorithm α_k, where this algorithm takes as additional parameters a description of (the adjacency matrix of) G. Define $\mathbf{U} = (u_1, 0, u_2, 0, \ldots, u_n, 0)$ and $\mathbf{V} = (v_1, 0, v_2, 0, \ldots, v_n, 0)$.

Suppose that there is an edge from $\mathbf{x}$ to $\mathbf{y}$ in $\mathcal{C}_k(G)$ where $y_i \neq x_i$, for some $i \in \{1, 2, \ldots, n\}$ (in particular, both $\mathbf{x}$ and $\mathbf{y}$ are proper k-colourings of G). There is a path in the faulty Q_{2n}^k as follows:

$$\begin{aligned}
&(x_1, 0, \ldots, x_{i-1}, 0, x_i, 0, x_{i+1}, 0, \ldots, x_n, 0) \\
&\quad \rightarrow (x_1, 0, \ldots, x_{i-1}, 0, x_i, 1, x_{i+1}, 0, \ldots, x_n, 0) \\
&\quad \rightarrow (x_1, 0, \ldots, x_{i-1}, 0, x_i + 1, 1, x_{i+1}, 0, \ldots, x_n, 0) \\
&\quad \rightarrow \ldots \\
&\quad \rightarrow (x_1, 0, \ldots, x_{i-1}, 0, y_i, 1, x_{i+1}, 0, \ldots, x_n, 0) \\
&\quad \rightarrow (x_1, 0, \ldots, x_{i-1}, 0, y_i, 0, x_{i+1}, 0, \ldots, x_n, 0).
\end{aligned}$$

Thus, if there is a path from $\mathbf{u}$ to $\mathbf{v}$ in $\mathcal{C}_k(G)$ then there is a path from $\mathbf{U}$ to $\mathbf{V}$ in the faulty Q_{2n}^k. The converse follows similarly. Hence, there is a path in the graph $\mathcal{C}_k(G)$ from vertex $\mathbf{u}$ to vertex $\mathbf{v}$ if, and only if, the instance $(2n, k, \mathbf{U}, \mathbf{V}, G)$ of ST-CONN-$Q_n^k(\alpha_k)$ is a yes-instance. As the problem ST-CONN-$Q_n^k(\alpha_k)$ is clearly in **PSPACE**, the result follows from the **PSPACE**-completeness of k-COLOUR-PATH [1].

Now consider the case when $k = 3$. We can not proceed as above as 3-COLOUR-PATH is solvable in **P** [2]. Let G be a graph on n vertices and let $\mathbf{u}$ and $\mathbf{v}$ be proper 4-colourings of G. For $j \in \{0, 1, 2, 3\}$, define (a^j, b^j) (resp. $+(a^j, b^j), -(a^j, b^j)$) as:

$(0,0)$ (resp. $(0,1)$, $(1,0)$) if $j=0$;
$(0,1)$ (resp. $(1,1)$, $(0,0)$) if $j=1$;
$(1,1)$ (resp. $(1,0)$, $(0,1)$) if $j=2$;
$(1,0)$ (resp. $(0,0)$, $(1,1)$) if $j=3$.

Note that for all $j \in \{0,1,2,3\}$, $+(a^j,b^j) = (a^{j+1},b^{j+1})$ and $-(a^j,b^j) = (a^{j-1},b^{j-1})$.

For every $\mathbf{x} \in \{0,1,2,3\}^n$, make the following links of Q^3_{3n} healthy, for $i = 1,2,\ldots,n$, with all other links faulty:

$$(a^{x_1},b^{x_1},0,\ldots,a^{x_{i-1}},b^{x_{i-1}},0,a^{x_i},b^{x_i},0,a^{x_{i+1}},b^{x_{i+1}},0,\ldots,a^{x_n},b^{x_n},0) \to$$
$$(a^{x_1},b^{x_1},0,\ldots,a^{x_{i-1}},b^{x_{i-1}},0,a^{x_i},b^{x_i},1,a^{x_{i+1}},b^{x_{i+1}},0,\ldots,a^{x_n},b^{x_n},0)$$
if $G(\mathbf{x})$ is a proper colouring;
$$(a^{x_1},b^{x_1},0,\ldots,a^{x_{i-1}},b^{x_{i-1}},0,a^{x_i},b^{x_i},1,a^{x_{i+1}},b^{x_{i+1}},0,\ldots,a^{x_n},b^{x_n},0) \to$$
$$(a^{x_1},b^{x_1},0,\ldots,a^{x_{i-1}},b^{x_{i-1}},0,\pm(a^{x_i},b^{x_i}),1,a^{x_{i+1}},b^{x_{i+1}},0,\ldots,a^{x_n},b^{x_n},0);$$
$$(a^{x_1},b^{x_1},0,\ldots,a^{x_{i-1}},b^{x_{i-1}},0,a^{x_i},b^{x_i},1,a^{x_{i+1}},b^{x_{i+1}},0,\ldots,a^{x_n},b^{x_n},0) \to$$
$$(a^{x_1},b^{x_1},0,\ldots,a^{x_{i-1}},b^{x_{i-1}},0,a^{x_i},b^{x_i},0,a^{x_{i+1}},b^{x_{i+1}},0,\ldots,a^{x_n},b^{x_n},0)$$
if $G(\mathbf{x})$ is a proper colouring.

It is trivial to verify that the resulting faulty 3-ary $3n$-cube can be described by a fault-describing algorithm α'_3 where this algorithm takes as additional parameters a description of (the adjacency matrix of) G. Define $\mathbf{U} = (a^{u_1},b^{u_1},0,a^{u_2},b^{u_2},0,\ldots,a^{u_n},b^{u_n},0)$ and $\mathbf{V} = (a^{v_1},b^{v_1},0,a^{v_2},b^{v_2},0,\ldots,a^{v_n},b^{v_n},0)$.

Suppose that there is an edge from $\mathbf{x}$ to $\mathbf{y}$ in $\mathcal{C}_4(G)$ where $y_i \neq x_i$, for some $i \in \{1,2,\ldots,n\}$ (in particular, both $\mathbf{x}$ and $\mathbf{y}$ are proper 4-colourings of G). There is a path in the faulty Q^3_{3n} as follows:

$$\begin{aligned}
&(a^{x_1},b^{x_1},0,\ldots,a^{x_{i-1}},b^{x_{i-1}},0,a^{x_i},b^{x_i},0,a^{x_{i+1}},b^{x_{i+1}},0,\ldots,a^{x_n},b^{x_n},0)\\
&\to (a^{x_1},b^{x_1},0,\ldots,a^{x_{i-1}},b^{x_{i-1}},0,a^{x_i},b^{x_i},1,a^{x_{i+1}},b^{x_{i+1}},0,\ldots,a^{x_n},b^{x_n},0)\\
&\to (a^{x_1},b^{x_1},0,\ldots,a^{x_{i-1}},b^{x_{i-1}},0,a^{x_i+1},b^{x_i+1},1,a^{x_{i+1}},b^{x_{i+1}},0,\\
&\qquad\qquad\qquad\ldots,a^{x_n},b^{x_n},0)\\
&\ldots\\
&\to (a^{x_1},b^{x_1},0,\ldots,a^{x_{i-1}},b^{x_{i-1}},0,a^{y_i},b^{y_i},1,a^{x_{i+1}},b^{x_{i+1}},0,\\
&\qquad\qquad\qquad\ldots,a^{x_n},b^{x_n},0)\\
&\to (a^{x_1},b^{x_1},0,\ldots,a^{x_{i-1}},b^{x_{i-1}},0,a^{y_i},b^{y_i},0,a^{x_{i+1}},b^{x_{i+1}},0,\ldots,a^{x_n},b^{x_n},0).
\end{aligned}$$

Thus, if there is a path from $\mathbf{u}$ to $\mathbf{v}$ in $\mathcal{C}_4(G)$ then there is a path from $\mathbf{U}$ to $\mathbf{V}$ in the faulty Q^3_{3n}. The converse follows similarly. Hence, there is a path in the graph $\mathcal{C}_4(G)$ from vertex $\mathbf{u}$ to vertex $\mathbf{v}$ if, and only if, the instance $(3n,3,\mathbf{U},\mathbf{V},G)$ of ST-CONN-$Q^k_n(\alpha'_3)$ is a yes-instance. As the problem ST-CONN-$Q^k_n(\alpha'_3)$ is clearly in **PSPACE**, the result follows from the **PSPACE**-completeness of 4-COLOUR-PATH [1]. □

It is trivial to verify that the reduction from 4-Colour-Path to st-Conn-$Q_n^k(\alpha'_3)$ in the second part of Theorem 1 actually builds an instance of the problem st-Conn-$Q_{3n}(\alpha'_3)$, and the reasoning there also applies to the faulty hypercube. Hence, we have the following result.

Theorem 2. There exists a fault-describing algorithm α so that the problem st-Conn-$Q_n(\alpha)$ is **PSPACE**-complete.

4. Specific routing algorithms

In this section, we look at the complexity of our connectivity problems when tied to specific routing algorithms. Suppose that we have some specific routing algorithm, like dimension-order routing; call our routing algorithm *Algorithm-X*. Algorithm-X results in a fault-describing algorithm α_X that details the healthy links emanating from some node. As we have detailed above, we also wish to supply some faults and to ascertain whether we can send a packet from the source to the destination, according to Algorithm-X, so as to avoid these faults. If β_X is a modification of α_X so that the healthy links resulting from β_X form a subset of the healthy links resulting from α_X (so, for example, β_X corresponds to dimension-order routing in the presence of additional faults) then we say that β_X is a *fault-describing Algorithm-X routing algorithm* and we denote an arbitrary fault-describing Algorithm-X routing algorithm by α_X^f (so, for example, we might have a fault-describing dimension-order routing algorithm).

Consider the following routing algorithm which is obtained by iterating dimension-order routing. In more detail, *iterated dimension-order routing* works as follows: for each dimension, we compute whether we need to increment or decrement the component in this dimension so as to obtain a shortest path from the source to the destination (if a shortest path can be obtained by either incrementing or decrementing in some dimension then we always increment) and we call this the direction of each dimension; given the current node, the destination node and the dimension order, we cycle through the dimensions, in order, until we find a dimension over which it is possible to move by incrementing or decrementing the component according to the direction of the dimension. This continues until we can progress no further or reach the destination. We denote the resulting fault-describing algorithm by α_{ido}.

Note that in the absence of any faults in Q_n^k, iterated dimension-order routing results in the identical path to that obtained when dimension-order routing is applied. However, in the presence of additional faults it may be the case that whilst standard dimension-order routing does not deliver a packet from the source to the destination, iterated dimension-order routing does. For example, suppose we attempt to send a packet from the source $(0, 0, 0)$ to the destination $(2, 2, 2)$ in Q_3^5 under dimension-order routing where the link $((0, 0, 0), (1, 0, 0))$ is the only fault and where the order of dimensions is $1, 2, 3$. The packet will not be routed across dimension 1 and so never leaves $(0, 0, 0)$. However, iterated dimension-order routing yields the following traversal for the packet:

$$(0,0,0) \to (0,1,0) \to (1,1,0) \to (2,1,0) \to (2,2,0) \to (2,2,1) \to (2,2,2).$$

As remarked earlier, dimension-order routing (in a faulty k-ary n-cube) results in a connectivity problem that can be solved in polynomial-time (as there is at most 1 path from the source to the destination and this path has length that is linear in n). Even when k is even and we allow non-determinism when some component of the source differs from the corresponding component of the destination (in the same dimension) by $\frac{k}{2}$, the resulting connectivity problem is still clearly solvable in polynomial-time. Of course, the connectivity problem corresponding to iterated dimension-order routing is still solvable in polynomial-time (as the path chosen from the source is always unique and has length at most $n\lfloor\frac{k}{2}\rfloor$). However, things change when we extend iterated dimension-order routing so that, when k is even, we allow non-determinism when some component in some dimension of the source differs from the corresponding component of the destination by $\frac{k}{2}$. Denote the resulting fault-describing algorithm by $\alpha_{\pm ido}$.

Theorem 3. For each even $k \geq 4$, there exists a fault-describing iterated dimension-order routing algorithm $\alpha^f_{\pm ido}$ such that the problem ST-CONN-$Q^k_n(\alpha^f_{\pm ido})$ is **NP**-complete.

Proof. Let $\mathcal{I}$ be an instance of the problem SATISFIABILITY in which there are n clauses involving the n Boolean variables $X_1, X_2, \ldots, X_n$. In the k-ary $(n+1)$-cube Q^k_{n+1}, make the following links healthy, for $i = 1, 2, \ldots, n$, with all other links faulty:

$$\begin{aligned}
&(x_1, x_2, \ldots, x_n, 0) \to (x_1, x_2, \ldots, x_{i-1}, x_i \pm 1, x_{i+1}, \ldots, x_n, 0)\\
&\quad \text{where } x_j \in \{1, k-1\}, \text{ for } j \in \{1, 2, \ldots, i-1\}, \text{ and } x_j = 0,\\
&\quad\ \text{for } j \in \{i, i+1, \ldots, n\};\\
&(x_1, x_2, \ldots, x_n, 0) \to (x_1, x_2, \ldots, x_n, 1)\\
&\quad \text{where } x_j \in \{1, k-1\}, \text{ for } j \in \{1, 2, \ldots, n\}, \text{ and the truth assignment } \pi\\
&\quad \text{obtained by setting } \pi(X_j) = T, \text{ if } x_j = 1, \text{ and } \pi(X_j) = F, \text{ if } x_j = k-1,\\
&\quad \text{for } j = 1, 2, \ldots, n, \text{ is a satisfying truth assignment of the instance } \mathcal{I};\\
&(x_1, x_2, \ldots, x_n, x_{n+1}) \to (x_1, x_2, \ldots, x_{i-1}, x_i + 1, x_{i+1}, \ldots, x_n, x_{n+1})\\
&\quad \text{where } 1 \leq x_{n+1} \leq \frac{k}{2} - 1 \text{ and } 1 \leq x_j \leq \frac{k}{2} - 1, \text{ for all } j = 1, 2, \ldots, n;\\
&(x_1, x_2, \ldots, x_n, x_{n+1}) \to (x_1, x_2, \ldots, x_{i-1}, x_i - 1, x_{i+1}, \ldots, x_n, x_{n+1})\\
&\quad \text{where } 1 \leq x_{n+1} \leq \frac{k}{2} - 1 \text{ and } \frac{k}{2} + 1 \leq x_j \leq k - 1, \text{ for all } j = 1, 2, \ldots, n;\\
&(\frac{k}{2}, \frac{k}{2}, \ldots, \frac{k}{2}, x_{n+1}) \to (\frac{k}{2}, \frac{k}{2}, \ldots, \frac{k}{2}, x_{n+1} + 1)\\
&\quad \text{where } 1 \leq x_{n+1} \leq \frac{k}{2} - 1.
\end{aligned}$$

Let the source in Q^k_{n+1} be $(0, 0, \ldots, 0)$ and the destination be $(\frac{k}{2}, \frac{k}{2}, \ldots, \frac{k}{2})$, and let the dimension-order be $1, 2, \ldots, n, n+1$. It is trivial to verify that the resulting faulty k-ary $(n+1)$-cube can be described by a fault-describing algorithm $\alpha^f_{\pm ido}$, so that the algorithm takes the dimension-order and a description of $\mathcal{I}$ as additional parameters.

Suppose that the instance $\mathcal{I}$ is satisfiable by the truth assignment π. For each Boolean variable X_i, if $\pi(X_i) = T$ then set $\epsilon_i = +1$, and if $\pi(X_i) = F$ then set $\epsilon_i = -1$. The path

$$(0, 0, \ldots, 0, 0) \to (\epsilon_1, 0, \ldots, 0, 0) \to (\epsilon_1, \epsilon_2, \ldots, 0, 0) \to (\epsilon_1, \epsilon_2, \ldots, \epsilon_n, 0)$$

is a legitimate traversal by a packet from the source $(0, 0, \ldots, 0)$ towards the destination $(\frac{k}{2}, \frac{k}{2}, \ldots, \frac{k}{2})$ in Q^k_{n+1} according to the iterated dimension-order routing algorithm. As the truth assignment π satisfies $\mathcal{I}$, the subsequent traversal to the node $(\epsilon_1, \epsilon_2, \ldots, \epsilon_n, 1)$ is legitimate too. Thereafter, the iterated dimension-order routing algorithm trivially moves the packet to the destination by decrementing or incrementing, as appropriate, the components in each dimension, in dimension-order $1, 2, \ldots, n, n+1$, until each takes the value $\frac{k}{2}$.

Conversely, suppose that the iterated dimension-order routing algorithm can move a packet from the source $(0, 0, \ldots, 0)$ to the destination $(\frac{k}{2}, \frac{k}{2}, \ldots, \frac{k}{2})$ in (the faulty) Q^k_{n+1}. There must exist on any such path a link from a node $(x_1, x_2, \ldots, x_n, 0)$ to a node $(x_1, x_2, \ldots, x_n, 1)$, where each $x_i \in \{1, k-1\}$, and in order for this link to be healthy the truth assignment corresponding to the n-tuple $(x_1, x_2, \ldots, x_n)$ must be a satisfying truth assignment of $\mathcal{I}$. As the problem ST-CONN-$Q^k_n(\alpha^f_{\pm ido})$ can trivially be solved in **NP**, the result follows. □

We now turn to another routing algorithm. The *negative-first routing algorithm* [5] for a k-ary n-cube Q^k_n with a given set of faulty links is defined as follows. There are two phases to this routing algorithm. In the first phase, a packet is routed by choosing (randomly or via some prescribed dimension order) some dimension whose component is different from the corresponding component of the destination node and decrementing this component if possible (that is, the corresponding link of Q^k_n is not faulty). The first phase ends when it is not possible to decrement the component of any chosen dimension. In the second phase, the packet is routed by choosing some dimension whose component is different from the corresponding component of the destination node and incrementing this component if possible. The second phase ends when it is not possible to increment the component of any chosen dimension. Clearly, we can have different variations of this negative-first routing algorithm, for example, depending upon whether we fix a dimension order or whether we randomly choose a dimension (and so do not work relative to some dimension order). Note that in both cases and irrespective of which links are faulty, the resulting route might not be minimal although it is easy to see that the length of any route taken is bounded by $n(2k-3)$. Note also that in the first variation of negative-first routing, the resulting routing algorithm is deterministic.

So far, we have equated a routing algorithm with a sub-digraph R consisting of the links that might be used by that routing algorithm. For negative-first routing, this poses a problem as the availability of a link depends upon which phase of the algorithm we are in; that is, we are dealing with a 'dynamic' sub-digraph rather that a 'static' sub-digraph R. However, we can simulate negative-first routing in Q_n^k by routing in a static sub-digraph R' of Q_{n+1}^k as follows. Suppose that we have a source node $\mathbf{u}$ and a destination node $\mathbf{v}$ in Q_n^k and that we wish to consider routing a packet from $\mathbf{u}$ to $\mathbf{v}$ via negative-first routing. We simulate this by routing a packet from the source node $\mathbf{u}'$ of Q_{n+1}^k to the destination node $\mathbf{v}'$, where $\mathbf{u}' = (\mathbf{u}, 0)$ and $\mathbf{v}' = (\mathbf{v}, 1)$, so that a link $(\mathbf{x}, \mathbf{y})$ of Q_n^k available for use when the packet is at the node $\mathbf{x}$ in phase 1 (resp. phase 2) of negative-first routing corresponds to the link $((\mathbf{x}, 0), (\mathbf{y}, 0))$ (resp. $((\mathbf{x}, 1), (\mathbf{y}, 1))$) of R', with links of the form $((\mathbf{x}, 0), (\mathbf{y}, 1))$ of R' being available when negative-first routing is at the node $\mathbf{x}$ and phase 2 is about to be entered into (that is, all potential 'negatively oriented' links from $\mathbf{x}$ in Q_n^k are faults).

We consider below *simulated randomized negative-first routing* in Q_{n+1}^k, denoted α_{rnf}, where a dimension, from the dimensions $\{1, 2, \ldots, n\}$, chosen at some node $\mathbf{x}$ (in any 'phase' and from those dimensions whose components of $\mathbf{x}$ differ from the corresponding components of the destination node) is chosen at random.

Theorem 4. For each $k \geq 3$, there exists a fault-describing simulated randomized negative-first routing algorithm α_{rnf}^f such that the problem ST-CONN-$Q_n^k(\alpha_{rnf}^f)$ is **NP**-complete.

Proof. Let $\mathcal{I}$ be an instance of the problem SATISFIABILITY in which there are n clauses involving the n Boolean variables $X_1, X_2, \ldots, X_n$. In the k-ary $(2n+1)$-cube Q_{2n+1}^k, make the following links healthy, for all $i = 1, 2, \ldots, n$, with all other links faulty:

$$\begin{aligned}
&(x_1, x_2, x_3, x_4, \ldots, x_{2n-1}, x_{2n}, 0) \rightarrow (x'_1, x'_2, x'_3, x'_4, \ldots, x'_{2n-1}, x'_{2n}, 0)\\
&\quad \text{where } \{x_{2i-1}, x_{2i}\} = \{1\}, \{x'_{2i-1}, x'_{2i}\} = \{0, 1\}, \text{ and } x'_j = x_j, \text{ for}\\
&\quad j \in \{1, 2, \ldots, 2n\} \setminus \{2i-1, 2i\};\\
&(x_1, x_2, x_3, x_4, \ldots, x_{2n-1}, x_{2n}, 0) \rightarrow (x'_1, x'_2, x'_3, x'_4, \ldots, x'_{2n-1}, x'_{2n}, 1)\\
&\quad \text{where } \{x_{2j-1}, x_{2j}\} = \{0, 1\}, \text{ for } j \in \{1, 2, \ldots, n\}, \text{ and the truth assignment}\\
&\quad \pi \text{ obtained by setting } \pi(X_j) = T, \text{ if } x_{2j-1} = 1, \text{ and } \pi(X_j) = F, \text{ if } x_{2j} = 1,\\
&\quad \text{for } j = 1, 2, \ldots, n, \text{ is a satisfying truth assignment of the instance } \mathcal{I};\\
&(x_1, x_2, x_3, x_4, \ldots, x_{2n-1}, x_{2n}, 1) \rightarrow (x'_1, x'_2, x'_3, x'_4, \ldots, x'_{2n-1}, x'_{2n}, 1)\\
&\quad \text{where } x'_i = x_i + 1 \text{ and } x'_j = x_j, \text{ for } j \in \{1, 2, \ldots, 2n\} \setminus \{i\}.
\end{aligned}$$

Let the source in Q_{2n+1}^k be $(1, 1, \ldots, 1, 0)$ and the destination be $(1, 1, \ldots, 1, 1)$. It is trivial to verify that the resulting faulty k-ary $(2n+1)$-cube can be described by a fault-describing algorithm α_{rnf}^f, so that the algorithm takes a description of $\mathcal{I}$ as additional parameters. Moreover, arguing as we have done in previous proofs yields that the problem ST-CONN-$Q_n^k(\alpha_{rnf}^f)$ is **NP**-complete. □

5. Conclusions

In this note we have embarked upon the complexity-theoretic classification of routing algorithms in k-ary n-cubes and hypercubes in which there are faulty links. Our framework is relevant to the practical implementation of routing algorithms in interconnection networks and yields hardness results pertinent to the testing of routing algorithms. Moreover, our framework has been shown to satisfactorily dealing with both 'static' routing algorithms and 'dynamic' (phase-based) routing algorithms. The ease by which we obtain our completeness results exhibits the generic relevance of k-ary n-cubes and hypercubes in conjunction with basic and popular routing algorithms.

References

[1] P. Bonsma and L. Cereceda, Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances, *Theoretical Computer Science* 410 (50) (2009) 5215–5226.
[2] L. Cereceda, J. van den Heuvel and M. Johnson, Finding paths between 3-colourings, *Journal of Graph Theory*, to appear.
[3] W.J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, San Francisco, 2004.
[4] E.D. Demaine and S. Srinivas, Routing algorithms on static interconnection networks: a classification scheme, *International Journal of Computer Systems Science and Engineering* 12 (6) (1997) 359–367.
[5] C.J. Glass and L.M. Ni, The turn model for adaptive routing, *Proc. of* 19*th Ann. Int. Symp. on Computer Architecture* (*ISCA* '92), ACM Press (1992) 278–287.