Interconnection networks for parallel and distributed computing

Xiang, Yonghong

How to cite:
Xiang, Yonghong (2008) Interconnection networks for parallel and distributed computing, Durham theses, Durham University. Available at Durham E-Theses Online: http://etheses.dur.ac.uk/2156/

Use policy
The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a link is made to the metadata record in Durham E-Theses
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

Please consult the full Durham E-Theses policy for further details.
Interconnection Networks for Parallel and Distributed Computing

Yonghong Xiang

Supervisor: Professor Iain Stewart and Professor Hajo Broersma

The copyright of this thesis rests with the author or the university to which it was submitted. No quotation from it, or information derived from it may be published without the prior written consent of the author or university, and any information derived from it should be acknowledged.

A thesis presented for the degree of
Doctor of Philosophy

Department of Computer Sciences
University of Durham
England
August 2008

19 DEC 2008
Dedicated to
My wife Yunli LIU
Interconnection Networks for Parallel and Distributed Computing

Yonghong XIANG

Submitted for the degree of Doctor of Philosophy
August 2008

Abstract

Parallel computers are generally either shared-memory machines or distributed-memory machines. There are currently technological limitations on shared-memory architectures and so parallel computers utilizing a large number of processors tend to be distributed-memory machines. We are concerned solely with distributed-memory multiprocessors. In such machines, the dominant factor inhibiting faster global computations is inter-processor communication. Communication is dependent upon the topology of the interconnection network, the routing mechanism, the flow control policy, and the method of switching. We are concerned with issues relating to the topology of the interconnection network.

The choice of how we connect processors in a distributed-memory multiprocessor is a fundamental design decision. There are numerous, often conflicting, considerations to bear in mind. However, there does not exist an interconnection network that is optimal on all counts and trade-offs have to be made. A multitude of interconnection networks have been proposed with each of these networks having some good (topological) properties and some not so good.

Existing noteworthy networks include trees, fat-trees, meshes, cube-connected cycles, butterflies, Möbius cubes, hypercubes, augmented cubes, k-ary n-cubes, twisted cubes, n-star graphs, (n,k)-star graphs, alternating group graphs, de Bruijn networks, and bubble-sort graphs, to name but a few.

We will mainly focus on k-ary n-cubes and (n,k)-star graphs in this thesis. Meanwhile, we propose a new interconnection network called augmented k-ary n-
cubes.

The following results are given in the thesis.

1. Let \( k \geq 4 \) be even and let \( n \geq 2 \). Consider a faulty \( k \)-ary \( n \)-cube \( Q^k_n \) in which the number of node faults \( f_n \) and the number of link faults \( f_e \) are such that \( f_n + f_e \leq 2n - 2 \). We prove that given any two healthy nodes \( s \) and \( e \) of \( Q^k_n \), there is a path from \( s \) to \( e \) of length at least \( k^n - 2f_n - 1 \) (resp. \( k^n - 2f_n - 2 \)) if the nodes \( s \) and \( e \) have different (resp. the same) parities (the parity of a node in \( Q^k_n \) is the sum modulo 2 of the elements in the \( n \)-tuple over 0, 1, \cdots, \( k - 1 \) representing the node). Our result is optimal in the sense that there are pairs of nodes and fault configurations for which these bounds cannot be improved, and it answers questions recently posed by Yang, Tan and Hsu, and by Fu. Furthermore, we extend known results, obtained by Kim and Park, for the case when \( n = 2 \).

2. We give precise solutions to problems posed by Wang, An, Pan, Wang and Qu and by Hsieh, Lin and Huang. In particular, we show that \( Q^k_n \) is bipanconnected and edge-bipancyclic, when \( k \geq 3 \) and \( n \geq 2 \), and we also show that when \( k \) is odd, \( Q^k_n \) is \( m \)-panconnected, for \( m = \frac{n(k-1)+2k-6}{2} \), and \((k-1)\)-pancyclic (these bounds are optimal). We introduce a path-shortening technique, called *progressive shortening*, and strengthen existing results, showing that when paths are formed using progressive shortening then these paths can be efficiently constructed and used to solve a problem relating to the distributed simulation of linear arrays and cycles in a parallel machine whose interconnection network is \( Q^k_n \), even in the presence of a faulty processor.

3. We define an interconnection network \( AQ_{n,k} \) which we call the augmented \( k \)-ary \( n \)-cube by extending a \( k \)-ary \( n \)-cube in a manner analogous to the existing extension of an \( n \)-dimensional hypercube to an \( n \)-dimensional augmented cube. We prove that the augmented \( k \)-ary \( n \)-cube \( AQ_{n,k} \) has a number of attractive properties (in the context of parallel computing). For example, we show that the augmented \( k \)-ary \( n \)-cube \( AQ_{n,k} \) is a Cayley graph (and so is vertex-symmetric); has connectivity \( 4n - 2 \), and is such that we can build a
set of $4n - 2$ mutually disjoint paths joining any two distinct vertices so that the path of maximal length has length at most $\max\{(n - 1)k - (n - 2), k + 7\}$; has diameter $\left[\frac{k}{3}\right] + \left[\frac{k-1}{3}\right]$, when $n = 2$; and has diameter at most $\frac{k}{4}(n + 1)$, for $n \geq 3$ and $k$ even, and at most $\frac{k}{4}(n + 1) + \frac{n}{4}$, for $n \geq 3$ and $k$ odd.

4. We present an algorithm which given a source node and a set of $n - 1$ target nodes in the $(n, k)$-star graph $S_{n,k}$, where all nodes are distinct, builds a collection of $n - 1$ node-disjoint paths, one from each target node to the source. The collection of paths output from the algorithm is such that each path has length at most $6k - 7$, and the algorithm has time complexity $O(k^3n^4)$.

**Keywords:** interconnection network, fault-tolerance, embedding, node-disjoint paths, bipanconnectivity, bipancyclicity, hamiltonicity, $k$-ary $n$-cube, augmented $k$-ary $n$-cube, $(n, k)$-star graph.
Declaration

The work in this thesis is based on research carried out at the Department of Computer Sciences, Durham University, England. No part of this thesis has been submitted elsewhere for any other degree or qualification and it's all the author's work unless referenced to the contrary in the text.

This research has been documented, in part, within the following publications:


Copyright © 2008 by Yonghong Xiang.

"The copyright of this thesis rests with the author. No quotations from it should be published without the author's prior written consent and information derived from it should be acknowledged".
Acknowledgements

Many thanks to my primary supervisor Professor Iain Stewart for all his patience, encouragement, and guidance during my Ph.D study. Thanks to my second supervisor Professor Hajo Broersma for every aspect of help and support during my study.

Thanks to all of my friends in or outside of the UK. Especially to Dr. Tong Li, who helped me like an elder brother in many aspects; Dr. Florent Madelaine, who helped with my research since early 2005; Dr. John Bailey, who helped with my study and my English. I enjoyed my stay in Durham and will miss the time with Dr. Keith Gallagher, Dr. Shengchao Qin, Miss. Barbara Froner, Mr. Minjie Sun, Mr. Xiaofeng Du, Mr. Qingzheng Zheng, and many other colleagues.

Thanks to my family for their continuous support my study in the UK. Especially to my wife, who gave up her nice job in China and joins me in the UK to support my Ph.D study. Special thanks to my mom, who came to the UK and helped us to look after my son Liang (Alexander) Xiang for half a year.

Thanks to EPSRC, ORSAS and the Department of Computer Science, Durham University for their financial support during my Ph.D study.
Contents

Abstract iii

Declaration vi

Acknowledgements vii

1 Introduction 1

1.1 Parallel and distributed computers ......................... 1
1.2 Some popular interconnection networks ..................... 2
  1.2.1 k-ary n-cube: an alternative to hypercube .............. 3
  1.2.2 (n, k)-star graph: an alternative to n-star graph ... 4
1.3 Paths and cycles ........................................... 4
  1.3.1 Paths and cycles in non-faulty interconnection networks 4
  1.3.2 Paths and cycles in faulty interconnection networks ... 6
1.4 Organization of the thesis .................................. 7

2 Basic Definitions and Basic Results 9

2.1 Some basic graph definitions ............................... 9
2.2 Definitions and properties of some interconnection networks 13
  2.2.1 Mesh, torus ........................................... 13
  2.2.2 Hypercube ............................................ 14
  2.2.3 k-ary n-cube ......................................... 15
  2.2.4 n-star graph ......................................... 16
  2.2.5 (n, k)-star graph .................................... 17
  2.2.6 n-dimensional augmented cube ........................ 18

viii
Contents

2.2.7 Section summary .............................................. 19
2.3 Related results ...................................................... 21
    2.3.1 Structures embedding in interconnection networks .......... 21
    2.3.2 Disjoint paths in interconnection networks ................. 23

3 Embedding long paths in $k$-ary $n$-cubes with faulty nodes and links 26
    3.1 Introduction .................................................... 26
    3.2 Basic definitions ................................................. 27
    3.3 The base case ..................................................... 28
    3.4 The inductive step ............................................... 52
    3.5 Conclusions ..................................................... 62

4 Bipanconnectivity and bicyclicity of $k$-ary $n$-cube 64
    4.1 Introduction ..................................................... 64
    4.2 Existing bipanconnectivity results .............................. 66
    4.3 The general case when $k$ is even ................................ 67
    4.4 The general case when $k$ is odd ................................ 73
    4.5 An application ................................................... 80
    4.6 Conclusions ..................................................... 81

5 Augmented $k$-ary $n$-cube 83
    5.1 Introduction ..................................................... 83
    5.2 Basic definitions ............................................... 85
    5.3 Symmetry .......................................................... 87
    5.4 Connectivity ..................................................... 89
        5.4.1 The base case of our induction ............................... 90
        5.4.2 The induction step ......................................... 93
    5.5 The diameter ..................................................... 108
    5.6 Conclusions ..................................................... 111

6 One-to-Many Node-Disjoint paths in $(n, k)$-star graph 114
    6.1 Introduction ..................................................... 114
    6.2 Basic definitions and lemmas .................................... 116
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3</td>
<td>The case for $k = 2$</td>
<td>118</td>
</tr>
<tr>
<td>6.4</td>
<td>Building node-disjoint paths</td>
<td>121</td>
</tr>
<tr>
<td>6.4.1</td>
<td>The basic algorithm</td>
<td>122</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Paths in $S_k$</td>
<td>123</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Paths in $S_1 \in \text{blocked} \cap \text{some}$</td>
<td>125</td>
</tr>
<tr>
<td>6.4.4</td>
<td>Paths in $S_1 \not\in \text{blocked} \cap \text{some}$</td>
<td>132</td>
</tr>
<tr>
<td>6.5</td>
<td>Path lengths and complexity</td>
<td>140</td>
</tr>
<tr>
<td>6.6</td>
<td>Conclusions</td>
<td>144</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion and future work</td>
<td>145</td>
</tr>
<tr>
<td>A</td>
<td>Source code: verify the base case of Theorem 3.3.3</td>
<td>165</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Example: (a) 4 x 4 mesh, (b) 4 x 4 torus ....................... 14
2.2 Example of hypercubes: (a) $Q_1$, (b) $Q_2$, (c) $Q_3$ ............ 15
2.3 Example of star graph: (a) 3-star, (b) 4-star and (c) (5,2)-star ..... 18
2.4 Three augmented cubes $AQ_1$, $AQ_2$ and $AQ_3$ .................. 19

3.1 Cases (a)-(d) when $k = 8$ ........................................ 31
3.2 Cases (h)-(j) when $k = 8$ ........................................ 32
3.3 Cases (k)-(o) when $k = 8$ ........................................ 33
3.4 Partitioned $Q^k_2$'s. .............................................. 37
3.5 The hamiltonian cycle $E_{0,0}$ in $rt(0,6)$ in $Q^7_2$. .................. 41
3.6 Joining $p_0(s,e)$ to the amended cycle $C$. ...................... 42
3.7 The construction in Case (b). ..................................... 57
3.8 The construction in Case (c). ..................................... 58
3.9 The constructions in Case (d) when $e \in Q_{1,1}$. .................... 59

4.1 Case (a) of Fig. 2 of [163] and its correction ..................... 67
4.2 Other cases from [163] and their corrections ..................... 68
4.3 The different cases when $d(u^a, v)$ is odd ....................... 70
4.4 The different cases when $d(u^a, v)$ is even ..................... 72
4.5 The different cases when $d(u^a, v)$ is even ..................... 74
4.6 The different cases when $d(u^a, v)$ is odd ..................... 76
4.7 The different cases when $d(u^a, v)$ is even ..................... 78
4.8 The different cases when $d(u^a, v)$ is odd ..................... 79

5.1 An augmented 5-ary 2-cube. .................................... 87
List of Figures

5.2 The 6 disjoint paths when \(0 < i < j\). ........................................ 91
5.3 The 6 disjoint paths when \(0 < i = j\). ........................................ 92
5.4 The 6 disjoint paths when \(i = 0\). ........................................ 94
5.5 The \(4n - 2\) disjoint paths in Case 1. ........................................ 95
5.6 The 6 disjoint paths in Sub-case 2.1. ........................................ 98
5.7 The amendments in Sub-case 2.2. ........................................ 98
5.8 The amendments in Sub-case 2.3. ........................................ 100
5.9 The paths in Sub-case 3.1. ........................................ 101
5.10 The paths in Sub-case 3.2. ........................................ 103
5.11 The paths in Sub-case 4.1. ........................................ 104
5.12 The paths in Sub-case 4.2. ........................................ 106
5.13 The paths in Sub-case 4.3. ........................................ 107

6.1 An illustration of different cases. ........................................ 121
6.2 Dealing with 'bad' target nodes. ........................................ 132
6.3 The tree \(\tau\) of procedure calls. ........................................ 143
List of Tables

2.1 Special $k$-ary $n$-cubes ............................................. 16
2.2 Key properties of some important interconnection networks. .... 20
Chapter 1

Introduction

1.1 Parallel and distributed computers

Parallel computers are generally either shared-memory machines or distributed-memory machines. There are currently technological limitations on shared-memory architectures and so parallel computers utilizing a large number of processors tend to be distributed-memory machines. We are concerned solely with distributed-memory multiprocessors. In such machines, the dominant factor inhibiting faster global computations is inter-processor communication. Communication is dependent upon the topology of the interconnection network (how the processors are joined to one another), the routing mechanism (how the paths along which data is transmitted between processors are determined), the flow control policy (how channels and buffers are allocated to packets as they travel along a path in the interconnection network), and the method of switching (the method by which a packet is moved in the interconnection network). We are concerned with issues relating to the topology of the interconnection network.

The choice of how we connect processors in a distributed-memory multiprocessor is a fundamental design decision. There are numerous, often conflicting, considerations to bear in mind. For instance, we would like our interconnection network to be symmetric (to make programming and analysis easier), have small diameter (to lessen message-passing latency), be recursively decomposable (to aid scalability), be highly connected (to improve fault-tolerance and reliability), be regular of low
1.2 Some popular interconnection networks

degree (to lessen communication overheads and design complexity), support rapid and easy inter-processor communication, support the simulation of other machines based on other topologies, and so on (note that a small diameter is desirable even when using wormhole switching, as wormhole switching only comes to the fore when dealing with larger packets). These properties all give rise to improved computational performance. However, there does not exist an interconnection network that is optimal on all counts and trade-offs have to be made. A multitude of interconnection networks have been proposed with each of these networks having some good (topological) properties and some not so good.

Existing noteworthy networks include trees, fat-trees, meshes, cube-connected cycles, butterflies, Möbius cubes, hypercubes, augmented cubes, k-ary n-cubes, twisted cubes, n-stars, (n,k)-stars, alternating group graphs, de Bruijn networks, and bubble-sort graphs, to name but a few. In the following section, we will introduce several popular networks.

1.2 Some popular interconnection networks

The architecture of an interconnection network is usually represented by a graph. We use graphs and networks interchangeably. A network is represented as an undirected graph in the thesis.

Interconnection topologies can be classified as either single-stage or multi-stage networks. Multi-stage networks, such as the omega network [107], connect system resources through multiple intermediate stages of crossbar switching devices. The performance of multi-stage type networks has been extensively studied in the literature [1-3,101,147]. Single-stage networks incorporate the processing devices within the network itself, allowing direct communication between processors. A single stage network has smaller average latency and is more fault tolerant in comparison with multi-stage networks of the same size [65]. As a result, single-stage networks are gaining in popularity and have been employed in many existing large scale computing systems [65].

We are only interested in single-stage networks, and now we will briefly introduce
some popular interconnection network topologies including hypercubes, $k$-ary $n$-cubes, $n$-star graphs and $(n,k)$-star graphs.

1.2.1 \textit{k-ary $n$-cube: an alternative to hypercube}

Perhaps the most popular interconnection topology is the hypercube $Q_n$, on account of its properties and its extremely elegant realization as a graph whose nodes are indexed with bit-strings of length $n$ and whose edges join nodes of Hamming distance 1 (such a realization immediately yields elementary yet optimal routing algorithms and key topological information). The hypercube has been used as the interconnection topology of a number of distributed-memory multiprocessors, such as the Cosmic Cube [141], the Ametek S/14 [15], the iPSC [48,49], the Ncube [26,49] and the CM-200 [27], and the properties of hypercubes relevant to parallel computing have been well studied.

However, every node of $Q_n$ has degree $n$, and, consequently, as $n$ increases so does the degree of every node, which is undesirable. Hence, given a collection of processors, if we wish to connect these processors in the topology of a hypercube then we have no choice as to the degree of the nodes of the resulting network. One method of circumventing this problem, so as to still retain a 'hypercube-like' interconnection network, is to build parallel computers so that the underlying topology is the $k$-ary $n$-cube $Q^k_n$. The $k$-ary $n$-cube is similar in essence to the hypercube (the nodes being indexed by bit-strings of length $n$ where there are $k$, as opposed to 2, different bits), but by a judicious choice of $k$ and $n$ we can include a large number of nodes yet keep the degree of each node low. The $k$-ary $n$-cube $Q^k_n$ has not been investigated to the same extent as the hypercube, but it has still been well studied. Machines whose underlying topology is based on a $k$-ary $n$-cube include the Mosaic [142], the iWARP [24], the J-machine [127], the Cray T3D [96], the Cray T3E [9], the SGI Origin and the IBM Blue Gene [64], and so on.
1.3. Paths and cycles

1.2.2 \((n,k)\)-star graph: an alternative to \(n\)-star graph

The \(n\)-star graph [4] is an attractive alternative to the hypercube \(Q_n\), and has significant advantages over \(Q_n\), such as a lower degree and a smaller diameter. However, a practical restriction is the number of nodes: \(n!\) for an \(n\)-star graph. Since there is a large gap between \(n!\) and \((n + 1)!\), one may face the choice of either too few or too many available nodes. The \((n,k)\)-star graph preserves many attractive properties of the \(n\)-star graph such as node symmetry, hierarchical structure, maximal fault tolerance, and simple shortest routing. What's more, the two parameters \(n\) and \(k\) can be tuned to make a suitable choice for the number of nodes in the network and for a degree/diameter trade-off. This allows more flexibility in designing networks than star graphs offer.

The definition and some basic properties of hypercubes, \(k\)-ary \(n\)-cubes, \(n\)-star graphs and \((n,k)\)-star graphs will be given in Chapter 2.

1.3 Paths and cycles

1.3.1 Paths and cycles in non-faulty interconnection networks

It is important for an interconnection network to efficiently route data among nodes. Efficient routing can be achieved by using node-disjoint paths. In what follows, we will use disjoint paths for node-disjoint paths. Routing by disjoint paths among nodes can not only avoid communication bottlenecks, and thus increase the efficiency of message transmission, but also provide alternative paths in case of node failures.

There are three well-known paradigms for the study of disjoint paths in interconnection networks. The node-to-node (one-to-one) disjoint paths that constructs the maximal number of disjoint paths in the network between two given nodes. The node-to-set (one-to-many or many-to-one) disjoint paths that constructs disjoint paths in the network from a given node to each of the nodes in a given set (it is true that \(k\) disjoint paths exist for the node-to-set disjoint paths problem in a \(k\)-connected graph [123]). The \(k\)-pairwise disjoint paths (set-to-set disjoint paths or
1.3. Paths and cycles

many-to-many disjoint paths) that constructs \( k \) disjoint paths between the given \( k \) node-pairs.

Linear arrays (paths) and rings (cycles), which are two of the most fundamental networks for parallel and distributed computation, are suitable for designing simple algorithms with low communication costs. Numerous efficient algorithms designed on linear arrays and rings for solving various algebraic problems and graph problems can be found in [7,128]. Linear arrays and rings can also be used as control/data flow structures for distributed computation in arbitrary networks. For example, having a collection of processors connected in a ring means that all-to-all message passing can be undertaken by “daisy-chaining” messages around the ring. An application of longest paths to a practical problem was encountered in the on-line optimization of a complex Flexible Manufacturing System (see [10]). These applications motivate the embedding of paths and cycles in networks.

One important property relevant to parallel computing is hamiltonicity, for the existence of hamiltonian cycles in networks is of crucial importance, given the ubiquity of such cycles as data structures in many distributed algorithms (they are primarily used to facilitate message-passing). Not only is the existence of hamiltonian cycles of great importance but also the existence of hamiltonian paths, and more generally the existence of cycles and paths of different lengths. The existence of hamiltonian (or, at least, long) paths is extremely useful as we regularly need to simulate linear-array computations in distributed-memory multiprocessors; having a long path allows us to cater for such simulations where there are many different array lengths involved in the simulations. In addition, given the ubiquity of cycle-based computations and algorithms in parallel computations, not only is the simulation of linear-array-based computations important but so is the simulation of cycle-based computations (of varying lengths).

Other hamiltonicity-based algorithms are also important in interconnection networks, such as the existence of (almost-)hamiltonian path, hamiltonian connectivity, almost-hamiltonian-connectivity, \((m-)\)pancyclicity, \((m-)\)panconnectivity, and hamiltonian-laceability.
1.3.2 Paths and cycles in faulty interconnection networks

As more and more processors are incorporated into parallel machines, faults become more common, be it faults in the processors themselves or faults on the interprocessor connections. Given the significant cost of parallel machines, we would prefer to be able to tolerate small numbers of faults and still be able to use our parallel machine. A key property we would like our 'faulty' machine to have is that a large number of the healthy processors should remain in a connected component and be able to undertake significant parallel computations. However, we prefer that the (non-faulty portion of the) interconnection network remains connected.

A number of different contexts have been studied with respect to the existence of faults. For example, the existence of hamiltonian cycles, hamiltonian paths, cycles and paths of specific lengths, and so on, have been studied in a variety of interconnection networks where there are faulty nodes or links. In addition, other aspects of fault-tolerance have been considered with regard to broadcasting algorithms, Euler tour algorithms, wormhole routing algorithms, and so on.

Indeed, some parallel applications, such as those in image and signal processing, are originally designed for a cycle architecture, and it is important to have effective cycle embeddings in a network. Faults can be static or dynamic, and there are possibilities of faulty nodes, faulty links or both faulty nodes and links.

When we consider how many faults we can tolerate in a given context, there are often pathological situations which immediately yield upper bounds. However, it has been shown that for certain topologies and situations, the probability of such situations is extremely small and discounting them can yield a meaningful and improved analysis. For example, consider a $k$-ary $n$-cube where we wish to determine the maximum number of faulty nodes so that regardless of the distribution of these faults, the healthy nodes remain connected. Immediately we see that there are configurations of $2n$ faulty nodes (where all faulty nodes are adjacent to some given node) which disconnect the network. However, if one assumes that the distribution of faults is such that all nodes are incident with at least 1 healthy node then a $k$-ary $n$-cube can tolerate $4n - 3$ faulty nodes such that the healthy nodes remain connected [44] (this result is optimal). Similar results regarding the conditional fault
1.4. Organization of the thesis

connectivity of other networks have been obtained, e.g., for hypercubes [36,52]; for cube-connected cycles, undirected de Bruijn networks and Kautz networks [133]; and for twisted-cubes, crossed-cubes, M"obius cubes, star graphs, pancake graphs recursive circulant graphs, and $k$-ary $n$-cubes [36]. Related studies of the diameter of faulty networks, under similar conditional fault assumptions, have been undertaken for hypercubes [104] and star graphs [138]. Conditional fault assumptions have also been made and studied in the context of hamiltonian cycles for hypercubes [28], crossed-cubes [91], star graphs [61] and so on.

Some related work has also been done for meshes [165], torus networks [145], arrangement graphs [79], line digraph interconnection networks [161], multi-stage interconnection networks [160], pancake graphs [90], double loop networks [152], (binary) wrapped butterfly graphs [17], folded hypercubes [162], $(n,k)$-star graphs [86], Josephus cubes [117], gamma interconnection networks [35], twisted cubes [59,89], recursive circulant graphs [130], flexible hypercubes [95], de Bruijn networks [126], and so on. Note that the general problem of deciding whether a given hypercube or a $k$-ary $n$-cube with an arbitrary collection of faults has a hamiltonian cycle (where no conditional assumptions on the distribution or number of faults are made) is known to be NP-complete [14,28].

1.4 Organization of the thesis

This thesis is focused on four aspects research of interconnection networks.

In Chapter 2, some basic graph definitions will be given. Then we will introduce several popular interconnection networks including the definitions and some of their basic properties. Also, some related results will be given in this chapter.

In Chapter 3, we will consider embedding long paths in a $k$-ary $n$-cube $Q^k_n$ with faulty nodes and links. We will answer questions recently posed by Yang, Tan and Hsu [171], and by Fu [60]. Furthermore, we extend known results, obtained by Kim and Park [98], for the case when $n = 2$.

In Chapter 4, we will investigate the hamiltonian, pancyclic, panconnected, bipancyclic and bipanconnected properties of $k$-ary $n$-cubes. Precise solutions will be
1.4. Organization of the thesis

given to problems posed by Wang, An, Pan, Wang and Qu [163] and by Hsieh, Lin and Huang [82]. A path-shortening technique, called progressive shortening, will be introduced. We will strengthen existing results, showing that when paths are formed using progressive shortening then these paths can be efficiently constructed and used to solve a problem relating to the distributed simulation of linear arrays and cycles in a parallel machine whose interconnection network is $Q_n^k$, even in the presence of a faulty processor.

In Chapter 5, we will propose a new interconnection network called the augmented $k$-ary $n$-cube $AQ_{n,k}$. Some basic properties including degree, diameter, connectivity and one-to-one node-disjoint paths will be given for $AQ_{n,k}$.

In Chapter 6, we will present an algorithm which given a source node and a set of $n - 1$ target nodes in the $(n, k)$-star graph $S_{n,k}$, where all nodes are distinct, builds a collection of $n - 1$ node-disjoint paths, one from each target node to the source. The collection of paths output from the algorithm is such that each path has length at most $6k - 7$, and the algorithm has time complexity $O(k^3n^4)$.

Finally, Chapter 7 concludes the thesis and gives some future research topics.

Based on the recursive structural properties of $k$-ary $n$-cubes, augmented $k$-ary $n$-cubes and $(n, k)$-star graphs, we mainly use induction proof method in Chapter 3, 4, 5 and 6.
Chapter 2

Basic Definitions and Basic Results

In this chapter, we will introduce some basic graph definitions. Then we will introduce several popular interconnection networks and some of their basic properties.

2.1 Some basic graph definitions

Throughout the thesis, a network is represented as a loopless undirected graph. For graph theoretic definitions and notations, we follow [22].

\[ G = (V,E) \] is a graph if \( V \) is a finite set and \( E \) is a subset of \( \{(u,v) | (u,v) \text{ is an unordered pair of } V\} \). We say that \( V \) is the vertex (node) set and \( E \) is the edge (link) set. Two vertices \( u \) and \( v \) are adjacent if \( (u,v) \in E \). A graph \( H \) is called a subgraph of \( G \) if \( V(H) \subseteq V(G) \) and \( E(H) \subseteq E(G) \). In graph \( G \), the neighborhood of \( v \), denoted by \( N_G(v) \), is the set \( \{x | (v,x) \in E\} \). If it is clear which graph is considered, we write \( N(v) \) instead; the same holds for other notations using graphs as a subscript. The degree of a vertex \( v \), denoted by \( \deg(v) \), is the number of vertices in \( N(v) \). A graph \( G \) is \( k \)-regular if \( \deg(v) = k \) for every vertex \( v \in V \). A graph \( G \) is vertex-symmetric (or node-symmetric) if given any two distinct nodes \( v \) and \( v' \) of \( G \), there is an automorphism of \( G \) mapping \( v \) to \( v' \). A graph \( G = (V_0 \cup V_1, E) \) is bipartite if \( V(G) \) is the union of two disjoint sets \( V_0 \) and \( V_1 \) such that each edge consists of one vertex from each set; such a partition \((V_0, V_1)\) is called a bipartition.
of the graph. Given vertices \( u \) and \( v \), we say that \( u \) and \( v \) are in the same partite set if \( u, v \in V_i \) or in different partite sets if \( u \in V_i \) and \( v \in V_{1-i} \) for \( i \in \{0, 1\} \). A vertex cut of a graph \( G \) is a set \( S \subseteq V(G) \) such that \( G - S \) has more than one connected component. It is known that only complete graphs do not have vertex cuts. The connectivity or vertex-connectivity of \( G \), written \( \kappa(G) \), is defined as the minimum size of a vertex cut if \( G \) is not a complete graph, and \( \kappa(G) = |V(G)| - 1 \) otherwise. A graph \( G \) is called \( k \)-connected or \( k \)-vertex-connected if its vertex connectivity is \( k \) or greater. A graph \( G \) with vertex connectivity \( \kappa(G) \) can tolerate \( \kappa(G) - 1 \) node failures. This measure of fault tolerance, however, gives a poor indication about the impact of faults on the interconnection network. A more appropriate metric, which is often used for measuring the fault tolerance of a graph, is the fault-diameter, which is defined as the maximum diameter of any graph obtained from \( G \) by removing at most \( \kappa(G) - 1 \) nodes from \( G \) [100].

A path is a non-null sequence \( \rho = (v_1, e_2, v_3, e_3, v_4, \ldots, e_k, v_k) \) whose terms are alternately vertices and edges, such that, for \( 2 \leq i \leq k \), the ends of \( e_i \) are \( v_{i-1} \) and \( v_i \), and in which all the vertices \( v_1, v_2, \ldots, v_k \) are distinct. For convenience, we also write the path as \( \rho = (v_1, v_2, v_3, \ldots, v_k) \) or \( \rho = (v_1, e_2, e_3, \ldots, e_k, v_k) \). We also write the path \( \rho = (v_1, v_2, v_3, \ldots, v_k) \) as \( \rho = (v_1, \rho', v_i, v_{i+1}, \ldots, v_j, \rho'', v_l, \ldots, v_k) \), where \( \rho' \) is the path \( (v_1, v_2, v_3, \ldots, v_i) \) and \( \rho'' \) is the path \( (v_j, v_{j+1}, \ldots, v_l) \). We use \( \rho^{-1} \) to denote the path \( (v_k, v_{k-1}, \ldots, v_1) \). On occasion we might refer to a link \((x, y)\) as appearing on a path \( \rho(u, v) \), or equivalently the path \( \rho(u, v) \) as containing the link \((x, y)\); when we do, the notation denotes that if we traverse the path \( \rho(u, v) \) starting at node \( u \) then we shall reach node \( x \) immediately before we reach node \( y \). If \( \rho(u, v) \) is a path and \( x \) and \( y \) are nodes on this path then \( \rho(x, y) \) denotes the sub-path of \( \rho(u, v) \) starting at \( x \) and ending at \( y \). The length of a path \( \rho \) is the number of the edges in \( \rho \), denoted by \( |\rho| \). We use \( d_{G}(u, v) \) to denote the distance between \( u \) and \( v \) in graph \( G \), that is the length of the shortest path joining \( u \) and \( v \). The diameter of a graph \( G \), denoted by \( \text{dia}(G) \), is the greatest distance between any two vertices. A path is a hamiltonian path if its vertices are distinct and span \( V \). A cycle is a path with at least three vertices such that the first vertex is the same as the last vertex. A cycle is a hamiltonian cycle if it traverses every vertex of \( G \) exactly once.
A graph is *hamiltonian* if it has a hamiltonian cycle.

We say that a graph is hamiltonian-connected if there is an hamiltonian path joining any two distinct nodes of the graph. Note that any (non-trivial) bipartite graph cannot be hamiltonian-connected, though there might exist *almost-hamiltonian paths*, i.e., paths joining pairs of distinct nodes upon which all but one of the nodes of the graph appear (a solitary node not appearing on an almost-hamiltonian path is called the *residual node*). Irrespective of whether a graph is bipartite or not, we say that a graph is *almost-hamiltonian-connected* if there is a hamiltonian path of an almost-hamiltonian path joining any pair of distinct nodes. The concept of hamiltonian connectivity does not apply to bipartite graphs because bipartite graphs are definitely not hamiltonian connected except for a few exceptions such as $K_2$ or $K_1$. As such a property is important, the concept of hamiltonian laceability on bipartite graphs was introduced by Wong [164]. A bipartite graph $G = \{V_0 \cup V_1, E\}$ is *hamiltonian laceable* if there is a hamiltonian path between any two vertices $x$ and $y$ which are in different partite sets. A hamiltonian laceable graph $G$ is *k-edge-fault-tolerant hamiltonian laceable* if $G - E'$ is hamiltonian laceable where $E'$ is subset of $E$ with $|E'| \leq k$. On the condition of $|V_0| = |V_1|$, Hsieh et al. [80] proposed the concept of strong hamiltonian laceability. $G$ is *strongly hamiltonian laceable* if it is hamiltonian laceable and there is a path of length $|V_0| + |V_1| - 2$ between any two vertices in the same partite set. A strongly hamiltonian laceable graph $G$ is *k-edge-fault-tolerant strongly hamiltonian laceable* if $G - E'$ is strongly hamiltonian laceable where $E'$ is subset of $E$ with $|E'| \leq k$. Lewinter and Widulski [109] introduced another concept, hyper hamiltonian laceability. $G$ is *hyper hamiltonian laceable* if it is hamiltonian laceable and for any vertex $v \in V_i$, there is a hamiltonian path of $G \setminus \{v\}$ between any two vertices in $V_{1-i}$. A hyper-hamiltonian laceable graph $G$ is *k-edge-fault-tolerant hyper-hamiltonian laceable* if $G - E'$ is hamiltonian laceable where $E'$ is subset of $E$ with $|E'| \leq k$. So hyper hamiltonian laceability implies strong hamiltonian laceability.

A *k-container* of $G$ between $u$ and $v$, $C(u,v)$, is a set of $k$ internally disjoint paths between $u$ and $v$. A $k$-container $C(u,v)$ of $G$ is a *$k^*$-container* if it contains all vertices of $G$. A graph $G$ is *$k^*$-connected* if there exists a $k^*$-container between any
2.1. Some basic graph definitions

two distinct vertices. Obviously, a 1*-connected graph (there is a path connecting any two nodes and covering all the nodes in the graph) is a hamiltonian connected graph, and a 2*-connected graph (there are two disjoint paths between any two nodes, and these two paths cover all nodes in the graph; thus they form a cycle, and all nodes are on the cycle.) is a hamiltonian graph. The spanning connectivity of a graph $G$, $\kappa^*(G)$, is the largest integer $k$ such that $G$ is $w^*$-connected for all $1 \leq w \leq k$. A graph $G$ is super spanning connected if $\kappa^*(G) = \kappa(G)$.

The concept of pancyclicity was extended to vertex-pancyclicity by Hobbs [76] and edge-pancyclicity by Alspach and Hare [8]. Let $n = |V(G)|$. A graph $G$ is called vertex-pancyclic if for any vertex $u$, there exists a cycle of every length from 3 to $n$ containing $u$, and edge-pancyclic if for any edge $e$, there exists a cycle containing $e$ of every length from 3 to $n$. If we fix one edge (two linked vertices), there exist cycles of every length from 3 to $n$, then if we fix one of these vertices, the result will still hold. So, every edge-pancyclic graph is vertex-pancyclic. The graph $G$ is almost-pancyclic if it contains a cycle of every possible length between 4 and $n$, and bipancyclic if it contains a cycle of every possible even length between 4 and $n$ (the definition of bipancyclicity is intended primarily for bipartite graphs but can be applied to any graph). A graph $G$ is called edge-bipancyclic if every edge $e$ of $G$ lies on a cycle of every even length between 4 and $n$, and vertex-bipancyclic if every vertex $v$ of $G$ lies on a cycle of every even length between 4 and $n$. A graph $G$ is $k$-edge-fault-tolerant bipancyclic if the resulting graph by deleting any $k$ edges from $G$ is bipancyclic. A graph $G$ is $k$-edge-fault-tolerant edge-bipancyclic if the resulting graph by deleting any $k$ edges from $G$ is edge-bipancyclic. The graph $G$ is panconnected (resp. $m$-panconnected) if for any pair of distinct vertices $u$ and $v$, there is a path joining $u$ and $v$ of every length between $\text{dis}(u, v)$ (resp. $m > \text{dis}(u, v)$) and $n - 1$. The graph $G$ is bipanconnected if for any pair of distinct vertices $u$ and $v$, there is a path joining $u$ and $v$ of every length from $\{l : l = \text{dis}(u, v) + 2i, \text{where } 0 \leq i \leq \frac{n-\text{dis}(u,v)}{2}\}$.

The Hamming distance between two vectors $a$ and $b$ is the number of different positions in which $a$ and $b$ differ, denoted by $D_H(a, b)$. Let $a = a_n a_{n-1} \ldots a_1$ be an
2.2. Definitions and properties of some interconnection networks

An $n$-digit radix $k$ vector. The Lee weight of $a$ is defined as

$$W_L(a) = \sum_{i=1}^{n} |a_i|, \text{ where } |a_i| = \min(a_i, k - a_i).$$

The Lee distance between two vectors $a$ and $b$ is denoted by $D_L(a, b)$ and is defined to be $W_L(a - b)$. That is, the Lee distance between two vectors is the Lee weight of their bitwise difference, mod $k$.

For other graph theory definitions please refer to the bibliography.

2.2 Definitions and properties of some interconnection networks

We will define the mesh, torus, hypercube, $k$-ary $n$-cube, $n$-star graph and $(n, k)$-stav graph and their basic properties in this section.

2.2.1 Mesh, torus

Definition 2.2.1 An $n$-dimensional mesh system $M(s)$ consists of $s_1 \times s_2 \times \ldots \times s_n$ processors arranged in an $n$-dimensional grid. A processor in the grid is denoted by the coordinate $(x_1, x_2, \ldots, x_n)$, where $0 < x_i < s_i - 1$.

Specifically, we define a 2-dimensional mesh as follows.

Definition 2.2.2 An $m \times n$ (rectangular) mesh $M(m, n)$ is a graph of $m \times n$ nodes arranged in $m$ rows and $n$ columns, where the node lying in the $i$th row and $j$th column is identified with an ordered pair $(i, j)$, and two nodes $(i, j), (k, l)$ are adjacent if and only if either (a) $i = k, |j - l| = 1$, or (b) $j = l, |i - k| = 1$. An $m \times n$ mesh is a bipartite graph with the bipartition $(U_0, U_1)$, where

$$U_0 = \{(i, j) : 0 \leq i \leq m - 1, 0 \leq j \leq n - 1, i + j \text{ is even}\},$$

$$U_1 = \{(i, j) : 0 \leq i \leq m - 1, 0 \leq j \leq n - 1, i + j \text{ is odd}\}.$$

A 2-dimensional mesh $M(m, n)$ is also called a grid $Grid(m, n)$. 
2.2. Definitions and properties of some interconnection networks

The $n$-dimensional mesh network is currently one of the most popular topologies for massively parallel computer systems [151]. Low dimensional mesh networks, due to their low node degree, are more popular than the high dimensional mesh networks. The two-dimensional mesh topology has been adopted by Symult 2010 [140], Intel Touchstone DELTA [23] and Intel paragon [93]; the MIT J-machine [127] adopts three-dimensional mesh topology.

**Definition 2.2.3** A torus $T(m,n)$ is a mesh with wraparound edges in the rows and columns. A row-torus is a mesh with wraparound edges in the rows. The row-torus $rt(i,j)$ is the subgraph of $T(m,n)$ induced by the nodes on rows $i, i+1, \ldots, j$, if $i < j$, or rows $i, \ldots, m, 1, \ldots, j$, if $j < i$, but with all column links between nodes on row $j$ and nodes on row $i$ removed if $i = j + 1$ or $(i = 0$ and $j = k - 1)$.

Fig. 2.1(a) is an example of a $4 \times 4$ mesh, and Fig. 2.1(b) is an example of a $4 \times 4$ torus.

An $m \times n$ mesh with $m, n \geq 4$ is almost-hamiltonian-connected [92].

### 2.2.2 Hypercube

**Definition 2.2.4** The $n$-dimensional hypercube ($n$-cube) $Q_n$, for $n \geq 2$, has $2^n$ nodes indexed by $\{0, 1\}^n$, and there is a link $((u_n, u_{n-1}, \ldots, u_1), (v_n, v_{n-1}, \ldots, v_1))$ if, and only if, there exists $d \in \{1,2,\ldots,n\}$ such that $|u_d - v_d| = 1$, and $u_i = v_i$, for every $i \in \{1,2,\ldots,n\}\setminus\{d\}$.

Fig. 2.2(a), (b) and (c) depict $Q_1, Q_2$, and $Q_3$ respectively. The hypercube has been used as the interconnection topology of a number of distributed memory...
2.2. Definitions and properties of some interconnection networks

multiprocessors, such as the Cosmic Cube [141], the Ametek S/14 [15], the iPSC [48, 49], the Ncube [26, 49] and the CM-200 [27], and the properties of hypercubes relevant to parallel computing have been well studied. The n-cube is a connected graph of diameter n, and is regular of degree n [139]. The hypercube is a bipartite graph [108, 139]. In n-cube, the minimum distance between the nodes u and v is equal to the number of bits that differ between u and v, i.e., to the Hamming distance $D_H(u, v)$ [139].

2.2.3 $k$-ary n-cube

The hypercube $Q_n$ is a very popular interconnection topology on account of its properties and its extremely elegant realization as a graph. However, the node degree of $Q_n$ increases too fast, which is undesirable. Hence, a hypercube-like interconnection network $k$-ary $n$-cube $Q_n^k$ was proposed, as in $Q_n^k$, we can include a large number of nodes yet keep the degree of each node low by tuning $k$ and $n$.

**Definition 2.2.5** The $k$-ary $n$-cube $Q_n^k$, for $k \geq 1$ and $n \geq 1$, has $k^n$ nodes indexed by $\{0, 1, \ldots, k - 1\}^n$, and there is a link $((u_n, u_{n-1}, \ldots, u_1), (v_n, v_{n-1}, \ldots, v_1))$ if, and only if, there exists $d \in \{1, 2, \ldots, n\}$ such that $\min\{|u_d - v_d|, k - |u_d - v_d|\} = 1$, and $u_i = v_i$, for every $i \in \{1, 2, \ldots, n\} \setminus \{d\}$, and we say this is an edge of dimension $i$.

An index $d \in \{1, 2, \ldots, n\}$ is often referred to as a *dimension*. We can partition $Q_n^k$ over dimension $d$ by fixing the $d$th element of any node tuple at some value $v$, for every $v \in \{0, 1, \ldots, k - 1\}$. Such a partition proves to be extremely useful (in proofs by induction, as we shall see for example in Chapter 3 and Chapter 4).
2.2. Definitions and properties of some interconnection networks

The class of $k$-ary $n$-cubes contains as special cases many topologies important to parallel processing, such as rings, hypercubes, and tori. Table 2.1 summarizes the special cases of $k$-ary $n$-cubes [120]. Fig. 2.1(b) is a $4 \times 4$ torus, and is also a 4-ary 2-cube.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>$\geq 3$</td>
</tr>
<tr>
<td>2</td>
<td>Point (cycle)</td>
</tr>
<tr>
<td>2</td>
<td>Point (torus)</td>
</tr>
<tr>
<td>2</td>
<td>Point</td>
</tr>
<tr>
<td>$\geq 3$</td>
<td>Edge (hypercube/cycle)</td>
</tr>
<tr>
<td>$\geq 3$</td>
<td>Square (hypercube/torus)</td>
</tr>
<tr>
<td>$\geq 3$</td>
<td>Hypercube</td>
</tr>
<tr>
<td>$\geq 3$</td>
<td>Ring</td>
</tr>
<tr>
<td>$\geq 3$</td>
<td>Torus</td>
</tr>
<tr>
<td>$\geq 3$</td>
<td>$k$-ary $n$-cube</td>
</tr>
</tbody>
</table>

Table 2.1: Special $k$-ary $n$-cubes

We now give some basic properties of $k$-ary $n$-cubes. A $k$-ary $n$-cube is a regular graph. The degree of each node is $n$ for $k = 2$ and $2n$ for $k \geq 3$. The number of edges in a $k$-ary $n$-cube is $nk^{n-1}$ for $k = 2$ and $nk^n$ for $k \geq 3$ [120]. $\text{dia}(Q_n^k) = n \lfloor \frac{k}{2} \rfloor$ [25]. In $Q_n^k$, the length of a shortest path between any two nodes is equal to their Lee distance [25]. $Q_2^k$ is bipancyclical, bipancyclic, almost-hamilton-connected, and if $k$ is odd, $Q_2^k$ is hamilton-connected, and $Q_n^k$ is almost-hamilton-connected, and hamilton-connected if $k$ is odd [163]. $Q_n^k$ is node-symmetric [98]. A $k$-ary $n$-cube contains $k$ composite subcubes, each of which is a $k$-ary $(n-1)$-cube, and the number of edges with endpoints in different composite subcubes is $k^{n-1}$ for $k = 2$ and $k^n$ for $k \geq 3$ [120].

2.2.4 $n$-star graph

The $n$-star graph [4] is an attractive alternative to the $n$-cube, as it has significant advantages over the $n$-cube, such as a lower degree and a smaller diameter.

**Definition 2.2.6** The $n$-star graph $S_n$ has node set $V(S_n) = \{(u_1, u_2, \ldots, u_n) : \text{each } u_i \in \{1, 2, \ldots, n\} \text{ and } u_i \neq u_j, \text{ for } i \neq j\}$, and there is an edge $((u_1, u_2, \ldots, u_n), (v_1, v_2, \ldots, v_n))$ if, and only if, $u_1 = v_i$ and $u_i = v_1$, for some $i \in \{2, 3, \ldots, n\}$, with $u_l = v_l$, for all $l \in \{2, 3, \ldots, n\} \setminus \{i\}$.
2.2. Definitions and properties of some interconnection networks

Fig. 2.3(a) shows a 3-star graph, (b) shows a 4-star graph. The \( n \)-star graph \( S_n \), has \( n! \) nodes, and \( \frac{n! \times (n-1)!}{2} \) edges. It is regular of degree \((n - 1)\) and has diameter \( \text{dia}(S_n) = \frac{3(n-1)}{2} \). \( S_n \) is node and edge symmetric and is \((n - 1)\)-connected [4, 43].

The star graph, which belongs to the class of Cayley graphs [5], possesses many nice topological properties such as recursiveness, symmetry, maximal fault tolerance, sublogarithmic degree and diameter, and strong resilience [5], which are all desirable when we are designing the interconnection topology for a parallel and distributed system. Besides, the star graph can embed rings [135], meshes [137], trees [16], and hypercubes [125]. Many efficient algorithms [7] have been designed on the star graph.

The star graph has been extensively studied. Its topological properties have been analyzed in [43, 135, 153]. Many efficient communication algorithms for shortest-path routing [135], multiple-path routing [43], broadcasting [122], gossiping [18], and scattering [57] were proposed. Many efficient algorithms have been designed for sorting and merging [124], selection [135], Fourier transform [56], and computational geometry [6].

2.2.5 \((n, k)\)-star graph

In order to avoid the significant jump from \( n! \) nodes in an \( n \)-star graph to \((n + 1)!\) nodes in an \((n + 1)\)-star graph, \((n, k)\)-star graphs were devised, as 'generalized' \( n \)-star graphs.

**Definition 2.2.7** Let \( n > k \geq 1 \). The \((n, k)\)-star graph, denoted \( S_{n,k} \), has node set \( V(S_{n,k}) = \left\{(u_1, u_2, \ldots, u_k) : \text{each } u_i \in \{1, 2, \ldots, n\} \text{ and } u_i \neq u_j, \text{ for } i \neq j\right\} \), and there is an edge \( ((u_1, u_2, \ldots, u_k), (v_1, v_2, \ldots, v_k)) \) if, and only if, either:

- \( u_i = v_i \), for \( 2 \leq i \leq k \), and \( u_1 \neq v_1 \) (a 1-edge);
- \( u_1 = v_1 \) and \( u_i = v_1 \), for some \( i \in \{2, 3, \ldots, k\} \), with \( u_l = v_l \), for all \( l \in \{2, 3, \ldots, k\} \setminus \{i\} \) (an \( i \)-edge).

In consequence, \( S_{n,k} \) has \( \frac{n!}{(n-k)!} \) nodes and \( \frac{n-1}{2} \times \frac{n!}{(n-k)!} \) edges. Note that \( S_{n,n-1} \) is isomorphic to the \( n \)-star \( S_n \), and that \( S_{n,1} \) is a clique on \( n \) nodes [39].
2.2. Definitions and properties of some interconnection networks

Fig. 2.3(c) shows a (5,2)-star graph.

Since their introduction in [39], \((n, k)-\)star graphs have been well-studied and their basic topological and algorithmic properties are well-understood. For example:

The diameter \(\text{dia}(S_{n,k})\) of \(S_{n,k}\) is given by

\[
\text{dia}(S_{n,k}) = \begin{cases} 
2k - 1 & \text{if } 1 \leq k \leq \lfloor \frac{n}{2} \rfloor, \\
n - 1 - \lfloor \frac{n}{2} \rfloor & \text{if } \lfloor \frac{n}{2} \rfloor + 1 \leq k \leq n - 1
\end{cases}
\]

in [39]. The \((n, k)-\)star graphs form a hierarchical family of graphs, each of which is node-symmetric [39]; they can be recursively decomposed in a number of ways [39]; they have a simple shortest-path routing algorithm [39]; the node-connectivity of \(S_{n,k}\) is \(n - 1\) [38]; and their fault-diameters are at most their fault-free-diameters plus 3 [38]. Let \(S_{n-1,k-1}(i)\) denote a subgraph of \(S_{n,k}\) induced by all the nodes with the same last symbol \(i\), for some \(1 \leq i \leq n\). \(S_{n,k}\) can be decomposed into \(n\) subgraphs \(S_{n-1,k-1}(i), 1 \leq i \leq n\), and each subgraph \(S_{n-1,k-1}(i)\) is isomorphic to \(S_{n-1,k-1}\) [39].

2.2.6 \(n\)-dimensional augmented cube

Several variations of hypercubes have been proposed and investigated to improve the efficiency of hypercube networks. Like the twisted cube [30], folded cube [51] or crossed cube [29], the augmented cube is one of the variations of hypercubes, which is proposed in [41] by Choudum and Sunithda.
2.2. Definitions and properties of some interconnection networks

Definition 2.2.8 Let \( n \geq 1 \) be an integer. The graph of the \( n \)-dimensional augmented cube, denoted by \( AQ_n \), has \( 2^n \) vertices indexed by \( \{0,1\}^n \). \( AQ_1 \) is the graph \( K_2 \) with vertex set \( \{0,1\} \). For \( n \geq 2 \), \( AQ_n \) can be recursively constructed by two copies of \( AQ_{n-1} \), denoted by \( AQ_n \{0\} \) and \( AQ_n \{1\} \) and by adding \( 2^n \) edges between \( AQ_{n-1} \{0\} \) and \( AQ_{n-1} \{1\} \) as follows:

Let the first bit of all nodes in \( AQ_{n-1} \{0\} \) (resp. in \( AQ_{n-1} \{1\} \)) be 0 (resp. 1). There is a link between node \( u = (0u_{n-1}u_{n-2} \ldots u_1) \) and \( v = (1v_{n-1}v_{n-2} \ldots v_1) \) if and only if either

(i) \( u_i = v_i \) for \( 2 \leq i \leq n \); in this case, \((u,v)\) is called a hypercube edge and we set \( v = u^h \), or

(ii) \( u_i = 1 - v_i \) for \( 2 \leq i \leq n \); in this case, \((u,v)\) is called a complement edge and we set \( v = u^c \).

Examples of augmented cubes \( AQ_1, AQ_2, \) and \( AQ_3 \) are shown in Fig. 2.4(a), (b) and (c) respectively.

The augmented cube of dimension \( n \) is a Cayley graph, \((2n-1)\)-regular, \((2n-1)\)-connected, and has diameter \( \lceil \frac{n}{2} \rceil \) \([41]\). It admits optimal routing and broadcasting algorithms that are similar to those for hypercubes and have the same time complexity \( O(n) \) \([41]\).

2.2.7 Section summary

We give the following table to summarize this section, which is a comprehensive version of Subsection 2.2.1 to 2.2.6.
Table 2.2: Key properties of some important interconnection networks.

<table>
<thead>
<tr>
<th>Network Name</th>
<th>Diameter</th>
<th>Degree</th>
<th>Nodes</th>
<th>Edges</th>
<th>Important Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mesh ((M(m, n)))</td>
<td>(m + n)</td>
<td>2, 3 or 4</td>
<td>(m \times n)</td>
<td>(2mn - m - n)</td>
<td>almost Hamilton-Connected</td>
</tr>
<tr>
<td>Torus ((T(m, n)))</td>
<td>(\max{\left\lfloor \frac{m}{2} \right\rfloor, \left\lfloor \frac{n}{2} \right\rfloor})</td>
<td>4</td>
<td>(m \times n)</td>
<td>(2mn)</td>
<td></td>
</tr>
<tr>
<td>Hypercube ((Q_n))</td>
<td>(n)</td>
<td>(n)</td>
<td>(2^n)</td>
<td>(n \times 2^{n-1})</td>
<td>Bipartite; (Dis(u, v) = D_H(u, v))</td>
</tr>
<tr>
<td>(k)-ary (n)-cube ((Q^k_n))</td>
<td>(\left\lfloor \frac{k}{2} \right\rfloor)</td>
<td>(n) if (k = 2)</td>
<td>(k^n)</td>
<td>(nk^{n-1}) if (k = 2)</td>
<td>(Q^k_2) is bipartite</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2n) if (k &gt; 2)</td>
<td>(nk^k) if (k \geq 3)</td>
<td></td>
<td>(Dis(u, v) = D_L(u, v))</td>
</tr>
<tr>
<td>Augmented Cube ((AQ_n))</td>
<td>(\left\lfloor \frac{n}{2} \right\rfloor)</td>
<td>(2n - 1)</td>
<td>(2^n)</td>
<td>((2n - 1)2^{n-1})</td>
<td>((2n - 1))-connected</td>
</tr>
<tr>
<td>(n)-star graph ((S_n))</td>
<td>(\frac{3(n-1)}{2})</td>
<td>(n - 1)</td>
<td>(n!)</td>
<td>(n!\frac{n-1}{2})</td>
<td>node- and edge-symmetric</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>((n - 1))-connected</td>
</tr>
<tr>
<td>((n, k))-star graph ((S_{n,k}))</td>
<td>(2k - 1) if (1 \leq k \leq \left\lfloor \frac{n}{2} \right\rfloor)</td>
<td>(n - 1)</td>
<td>(\frac{n!}{(n-k)!})</td>
<td>(\frac{n-1}{2} \times \frac{n!}{(n-k)!})</td>
<td>(S_{n,1}) is a clique; (S_{n,n-1} \cong S_n)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(S_{n,k}) is ((n - 1))-connected</td>
</tr>
</tbody>
</table>
2.3 Related results

In this section, we will mainly review the following two related topics in interconnection networks: structures embedding in interconnection networks, and node-disjoint paths problems in interconnection networks.

2.3.1 Structures embedding in interconnection networks

In some popular interconnection networks, we will consider the following properties: hamiltonicity, hamiltonian connectivity, ring/cycle embedding, (bi)panconnectivity, (bi)pancyclicity, and so on.

If there are faults in interconnection networks, we will only consider static faults.

**Definition 2.3.1** *Conditional fault assumption (CFA):* Each node is adjacent to at least two healthy (fault-free) nodes via healthy links.

Let $F_e$ denote the set of faulty edges (links) in the graph $G$, and let $F_v$ denote the set of faulty nodes in the graph. Let $f_e$ and $f_v$ denote the number of faulty edges and nodes respectively, i.e., $f_e = |F_e|$, and $f_v = |F_v|$.  

**Hypercube $Q_n$**

As the hypercube is a bipartite graph, there exist no odd cycles.

- If there are no faults, $Q_n$ is bipancyclic [139].
- If there are faulty edges in $Q_n$, the following results has been obtained.
  
The $Q_n$ is hamiltonian, if $f_e \leq n - 2$ [105], or under CFA and $f_e \leq 2n - 5$ [28].
  
The $Q_n$ is bipancyclic [154] under CFA and $f_e \leq 2n - 5$.
  
The $Q_n$ is proved to be edge-bipancyclic if
    
    - $n \geq 3$, and $f_e \leq n - 2$ by Li et al. [110], or
    
    - under CFA, $n \geq 4$ and $f_e \leq n - 1$ by Xu et al. [167], or
    
    - under CFA, $n \geq 4$ and $f_e \leq 2n - 5$ by Shih et al. [146].

Note that the minimum cycle length in [167] and [146] is 6.
2.3. Related results

- If there are node faults or both node and edge faults, the following results have been obtained.

It is proved that a longest cycle of length at least \(2^n - 2f_v\) can be embedded into \(Q_n\) if

- \(f_v \leq n - 2\) by Yang et al. [172], or
- \(f_e \leq n - 4, f_v \leq n - 1\) and \(f_e + f_v \leq n - 1\) by Tseng [157], or
- \(f_v > 0, f_e < n - 1\) and \(f_v + f_e \leq n - 1\) by Sengupta et al. [144], or
- \(f_v \leq 2n - 4\) by Fu et al. [58].

Tsai [155] proved that every fault-free edge of \(Q_n\), for \(n \geq 3\), lies on a fault-free cycle of every even length from 4 to \(2^n - 2f_v\) inclusive if \(f_e + f_v < n - 2\). Furthermore, he proved that \(Q_n\), for \(n \geq 5\), contains a fault-free cycle of every even length from 4 to \(2^n - 2f_v\) inclusive if \(f_e \leq n - 2\) and \(f_e + f_v \leq 2n - 4\).

**k-ary \(n\)-cube \(Q_n^k\)**

The k-ary \(n\)-cube \(Q_n^k\) is proved to be hamiltonian under different conditions:

- \(k \geq 3, n \geq 2\) and no faults [19,25]; or
- under CFA and \(f_e \leq 4n - 5\) [14]; or
- \(k \geq 3\) be an odd integer, and \(f_e + f_v \leq 2n - 2\) [171].

Let \(k \geq 3\) be an odd integer, if \(f_v + f_e \leq 2n - 3\), then the wounded \(k\)-ary \(n\)-cube is hamiltonian-connected [171].

**\(n\)-star graphs \(S_n\)**

If there are no faults in \(n\)-star graphs \(S_n\), then \(S_n\) is hamiltonian, bipancyclic and a variety of two- and multi-dimensional grids can be embedded into \(S_n\) [94].

\(S_n\) is hamiltonian if \(f_e \leq n - 3\) [158] or under CFA and \(f_e \leq 2n - 7, n \geq 4\) [61]. \(S_n\) is proved to be bipancyclic [111] and edge-bipancyclic [169] if \(f_e \leq n - 3\) and \(n \geq 3\), where the minimum cycle length is 6. The \(n\)-star graph is \((n - 3)\)-edge fault tolerant hamiltonian laceable, \((n - 3)\)-edge fault tolerant strongly hamiltonian laceable, and \((n - 4)\)-edge fault tolerant hyper hamiltonian laceable [113]. Tseng et
2.3. Related results

al. [158] found a cycle of length at least \( n! - 4f_v \), if \( f_v \leq n - 3 \). Hsieh et al. [81] found a path of length \( n! - 2f_v - 2(n! - 2f_v - 1) \) between two arbitrary vertices of even (odd) distance, if \( f_v \leq n - 5 \). Since \( S_n \) is bipartite with two partite sets of equal size, the path is longest for the worst-case scenario.

\((n, k)\)-star graphs \( S_{n,k} \)

Chang and Kim [32] found a cycle of length \( n!/(n - k)! - f_v \) in an \((n, k)\)-star graph when \( f_v \leq n - 3 \) and \( n - k = 2 \).

\( S_{n,k} \) is hamiltonian if \( f_v + k \leq n - 3 \), hamiltonian-connected if \( f_v + f_v \leq n - 4 \) [86]. \( S_{n,k} \) is super spanning connected if \( n \geq 3 \) and \((n - k) \geq 2 \) [87].

Chen et al. [37] showed that \( S_{n,k} \) is vertex-pancyclic when \( 1 \leq k \leq n - 4 \) and \( n \geq 6 \). Additionally, for \( n - 3 \leq k \leq n - 2 \), \( S_{n,k} \) is also vertex-pancyclic with the minimum cycle length is 6. Moreover, each constructed cycle in \( S_{n,k} \) can be made to contain a desired 1-edge.

2.3.2 Disjoint paths in interconnection networks

It is practically important to construct node-disjoint paths (disjoint paths for short) in networks, because they can be used to increase the transmission rate and enhance the transmission reliability. Besides, disjoint paths have applications in multi-path routing (such as Rabin’s information dispersal algorithm [136]), fault tolerance (see [47,60]), and communication protocols (see [85]).

We are only interested in disjoint paths problem in non-faulty interconnection networks. For more information about disjoint paths in faulty interconnection networks, please refer to [67,70,73,119].

One-to-one disjoint paths

Sets of one-to-one disjoint paths are also named containers.

1988 [139]: In the \( n \)-cube, let \( u \) and \( v \) be any two nodes and assume that \( D_H(u,v) < n \). Then there are \( D_H(u,v) \) disjoint paths of length \( D_H(u,v) \), and \( n \) disjoint paths of length at most \( D_H(u,v) + 2 \) between the nodes \( u \) and \( v \).

1997 [45]: Day and Al-Ayyoub constructed a set of \( n \) disjoint paths between any two nodes of a \( k \)-ary \( n \)-cube \( Q_n^k \). Each path is of length zero, two, or four.
plus the minimum length except for one path in a special case (when the Hamming distance between the two nodes is one) where the increase over the minimum length may attain eight. These results improve those obtained in [25] where the length of some of the paths has a variable increase (which can be arbitrarily large) over the minimum length.

2000 [150]: Su et al. showed that a set of \(d\) node disjoint paths is constructed between two arbitrary nodes of an incomplete WK-recursive network \(IK(d,t)\). The length is not greater than 2 times the diameter.

2002 [62]: Fu et al. constructed \(n+1\) disjoint paths between any two given nodes in \(n\)-dimensional Hierarchical Cubic Networks (HCS\((n))\), whose lengths are at most \(n + \lfloor \frac{n}{2} \rfloor + 3\). This improves on the containers of [40] whose lengths are \(2n + 6\) at most.

2005 [134]: Qiu and Akl gave an algorithm that finds \(n-1\) disjoint paths between any two nodes \(s\) and \(t\) in an \(n\)-star in optimal \(O(n^2)\) time such that no path has length more than \(\text{dis}(s,t) + 4\).

2007 [166]: Wu et al. found \(m+1\) disjoint paths between any two distinct nodes of an \(n\)-dimensional hierarchical hypercube network \(n\)-HHC network \((n = 2^m + m)\), whose lengths are not greater than \(\max\{\text{dia}(n\text{-HHC}) + 2m + 1, \text{dia}(n\text{-HHC}) + m + 4\}\), where \(\text{dia}(n\text{-HHC}) = 2^{m+1}\).

2008 [116]: Lin et al. described an algorithm for constructing a container of width \(n - 1\) between a pair of vertices in an \((n,k)\)-star graph with \(2 \leq k \leq n - 2\). The maximal path length is \(\text{dia}(S_{n,k}) + 2\) for \(2 \leq k \leq \lfloor \frac{n}{2} \rfloor\), or \(\text{dia}(S_{n,k})\) plus a value between 1 and 2 for \(\lfloor \frac{n}{2} \rfloor + 1 \leq k \leq n - 2\). The same problem for \((n,n-1)\)-star and \((n,1)\)-star graphs has been investigated in [115], where the lengths of the paths are at most \(\text{dis}(S_{n,n-1}) + 2\) and \(\text{dis}(S_{n,1}) + 1\) respectively.

One-to-many disjoint paths

1997 [71]: Gu and Peng gave an \(O(n^2)\) time algorithm, which finds \(n - 1\) disjoint paths of length at most \(\text{dia}(S_n) + 2\). (A lower bound on the length of the paths for the above problem in is \(\text{dia}(S_n) + 1\).)

1998 [106]: Latifi et al. computed the \(n\) vertex disjoint paths of length at most \(n + 1\) in a hypercube \(Q_n\) of dimension \(n\), given a source node and an arbitrary set of
at most \( n \) destination nodes; their algorithm is computationally simpler than that of [136].

**Many-to-many disjoint paths**

1996 [69]: Gu and Peng presented an algorithm for finding \( k \) disjoint paths where each path connects a pair of nodes from two given node sets in an \( n \)-cube \( Q_n \), where \( 1 \leq k \leq n \). The path length is at most \( n + \log k + 2 \), and the time complexity is \( O(kn \log^* k) \), where \( \log^* n = 0 \), if \( n \leq 1 \), and \( \log^* n = 1 + \log^* (\log n) \), if \( n > 1 \).

1998 [72]: Gu and Peng gave an many-to-many algorithm, which finds the \( k \) disjoint paths of length at most \( \text{dia}(S_n) + 5 \) in \( O(n^2) \) optimal time. This improves the previous results of \( 4(n - 2) \) (path length) and \( O(n^4 \log n) \) (time), respectively in [46].

2000 [68]: Given \( k \geq \left[ \frac{n}{2} \right] \) pairs of distinct nodes \( (s_1, t_1), \ldots, (s_k, t_k) \) in the \( n \)-cube \( Q_n \), Gu and Peng presented an algorithm finding the \( k \) disjoint paths with length at most \( n + \lfloor \log n \rfloor + 1 \) in \( O(n^2 \log^* n) \) time.

Up to now, we have given some results related to paths and cycles in interconnection networks. There are more related problems in this area, for example, the pairwise shortest path routing problem [66]. However, we are only interested in the above stated problems. In the next chapter, we will present an algorithm to show that there exists a long path in faulty \( k \)-ary \( n \)-cubes.
Chapter 3

Embedding long paths in $k$-ary $n$-cubes with faulty nodes and links

3.1 Introduction

In this chapter we study the existence of long paths and cycles in the presence of limited numbers of node and link faults in $k$-ary $n$-cubes. We are motivated by the work in four recent publications. In [98], Kim and Park study the existence of hamiltonian paths in two-dimensional tori. They provide conditions when a two-dimensional torus with at most 2 faulty nodes is hamiltonian, hamiltonian-connected and bi-hamiltonian-connected. In [60], Fu proves that an $n$-dimensional hypercube with $f_v \leq n-2$ is such that there is a path of length at least $2^n - 2f_v - \epsilon$ between any two distinct, healthy nodes, where $\epsilon = 1$ if the two nodes have different parities and $\epsilon = 2$ otherwise. In [77], Hsieh and Chang show that under CFA, Fu's result holds even when $f_v \leq 2n - 5$. In [171], Yang, Tan and Hsu prove that in a $k$-ary $n$-cube where $k$ is odd, if the number of faulty nodes and links is at most $2n - 3$ then there is a hamiltonian cycle, and if the number of faulty nodes and links is at most $2n - 2$ then there is a hamiltonian path joining any two, distinct healthy nodes. Note that Yang, Tan and Hsu prove no results when $k$ is even beyond remarking that when $k$ is even, the $k$-ary $n$-cube is bipartite and so if there is 1 faulty node then there can
be no hamiltonian cycle and there exists a pair of distinct, healthy nodes not joined by a hamiltonian path.

Our main result is as follows. Let \( k \geq 4 \) be even and let \( n \geq 2 \). In a faulty \( k \)-ary \( n \)-cube \( Q^k_n \) in which the number of node faults \( f_v \) and the number of link faults \( f_e \) are such that \( f_v + f_e \leq 2n - 2 \), given any two healthy nodes \( s \) and \( e \) of \( Q^k_n \), there is a path from \( s \) to \( e \) of length at least \( k^n - 2f_v - 1 \) (resp. \( k^n - 2f_v - 2 \)) if the nodes \( s \) and \( e \) have different (resp. the same) parities. Our result: resolves the situation in [171] when \( k \) is even; answers questions posed by Yang, Tan and Hsu, and by Fu; and extends known results, obtained by Kim and Park, for the case when \( n = 2 \).

The rest of this chapter is devoted to a proof by induction of our main theorem. Section 3.2 contains the basic definitions. In Section 3.3, we deal with the base case of the induction, and in Section 3.4, we deal with the inductive step. We present our conclusions in Section 3.5.

### 3.2 Basic definitions

Many structural properties of \( k \)-ary \( n \)-cubes are known, but of particular relevance for us is that a \( k \)-ary \( n \)-cube is node-symmetric. Throughout, we assume that addition on tuple elements is modulo \( k \).

We can partition \( Q^k_n \) over dimension \( d \). This results in \( k \) copies \( Q_{d,0}, Q_{d,1}, \ldots, Q_{d,k-1} \) of \( Q^k_{n-1} \), with corresponding nodes in \( Q_{d,0}, Q_{d,1}, \ldots, Q_{d,k-1} \) joined in a cycle of length \( k \) (in dimension \( d \)).

The parity of a node \( v = (v_n, v_{n-1}, \ldots, v_1) \) of \( Q^k_n \) is defined to be \( \sum_{i=1}^{n} v_i \) modulo 2. We speak of a node as being odd or even according to whether its parity is odd or even. A pair of nodes \( \{v, v'\} \) is odd (resp. even) if \( v \) and \( v' \) have different (resp. the same) parities.

We write paths in \( Q^k_n \) as sequences of incident links, and when \( k \) is even, paths necessarily consist of links joining, alternatively, odd and even nodes.

A fault in \( Q^k_n \) refers to a faulty node or a faulty link. If a node is faulty then we imagine that the node and its incident links do not exist; if a link is faulty then we imagine that this link does not exist. When we refer to a path in a faulty \( Q^k_n \), we
3.3. The base case

In this section, we deal with the base case of our forthcoming inductive proof of the main result, namely when we have a \( k \)-ary 2-cube with no more than 2 faults.

We consider \( Q^k_n \) as a \( k \times k \) grid with wrap-around and we think of a node \( v_{i,j} \) as indexed by its row \( i \) and column \( j \). Throughout, we assume that addition on row or column indices is modulo \( k \).

We define the following paths in the row-torus \( rt(0,1) \) (of some \( Q^k_5 \)). The names of these paths are derived from the shape of their pictorial representations (see the figures coming up). Also, if \( i = 0 \) then \( \tilde{i} = 1 \), and if \( i = 1 \) then \( \tilde{i} = 0 \).

\[
C^+_m(v_{i,j}, v_{i,j}) = (v_{i,j}, v_{i,j+1}), (v_{i,j+1}, v_{i,j+2}), \ldots, (v_{i,m-1}, v_{i,m}), (v_{i,m}, v_{i,m}), (v_{i,m-1}, v_{i,m-1}), (v_{i,m-2}, v_{i,m-2}), \ldots, (v_{i,j+1}, v_{i,j})
\]

where \( 0 \leq i \leq 1, 0 \leq j \leq k-1, 0 \leq m \leq k-1 \) and \( m \neq j \).
3.3. The base case

\[ C_m(v_{i,j}, v_{i,j}) = (v_{i,j}, v_{i,j-1}), (v_{i,j-1}, v_{i,j-2}), \ldots, (v_{i,m+1}, v_{i,m}), (v_{i,m}, v_{i,m+1}), (v_{i,m+1}, v_{i,m+2}), \ldots, (v_{i,j-1}, v_{i,j}) \]

where \( 0 \leq i \leq 1, 0 \leq j \leq k - 1, 0 \leq m \leq k - 1 \) and \( m \neq j \).

\[ N^+(v_{i,j}, v_{i,j'}) = (v_{i,j}, v_{i,j}), (v_{i,j}, v_{i,j+1}), (v_{i,j+1}, v_{i,j+1}), (v_{i,j+1}, v_{i,j+2}), (v_{i,j+2}, v_{i,j+2}), (v_{i,j+2}, v_{i,j+3}), (v_{i,j+3}, v_{i,j+3}), (v_{i,j+3}, v_{i,j+4}), \ldots, (v_{i,j'}, v_{i,j'}) \]

where \( 0 \leq i \leq 1, 0 \leq j \neq j' \leq k - 1 \) and \( |j - j'| \) is even.

\[ N^-(v_{i,j}, v_{i,j'}) = (v_{i,j}, v_{i,j}), (v_{i,j}, v_{i,j-1}), (v_{i,j-1}, v_{i,j-1}), (v_{i,j-1}, v_{i,j-2}), (v_{i,j-2}, v_{i,j-2}), (v_{i,j-2}, v_{i,j-3}), (v_{i,j-3}, v_{i,j-3}), (v_{i,j-3}, v_{i,j-4}), \ldots, (v_{i,j'}, v_{i,j'}) \]

where \( 0 \leq i \leq 1, 0 \leq j \neq j' \leq k - 1 \) and \( |j - j'| \) is even.

\[ Z^+(v_{i,j}, v_{i,j'}) = (v_{i,j}, v_{i,j+1}), (v_{i,j+1}, v_{i,j+1}), (v_{i,j+1}, v_{i,j+2}), (v_{i,j+2}, v_{i,j+2}), (v_{i,j+2}, v_{i,j+3}), (v_{i,j+3}, v_{i,j+3}), (v_{i,j+3}, v_{i,j+4}), \ldots, (v_{i,j'}, v_{i,j'}) \]

where \( 0 \leq i \leq 1, 1 \leq j \neq j' \leq k - 1 \) and \( |j - j'| \) is even.

\[ Z^-(v_{i,j}, v_{i,j'}) = (v_{i,j}, v_{i,j-1}), (v_{i,j-1}, v_{i,j-1}), (v_{i,j-1}, v_{i,j-2}), (v_{i,j-2}, v_{i,j-2}), (v_{i,j-2}, v_{i,j-3}), (v_{i,j-3}, v_{i,j-3}), (v_{i,j-3}, v_{i,j-4}), \ldots, (v_{i,j'}, v_{i,j'}) \]

where \( 0 \leq i \leq 1, 1 \leq j \neq j' \leq k - 1 \) and \( |j - j'| \) is even.

In addition, we define \( C_j^+(v_{i,j}, v_{i,j}) = C_j^-(v_{i,j}, v_{i,j}) = (v_{i,j}, v_{i,j}) \). We also use the above notation to describe paths in other row-tori of the form \( rt(l, l + 1) \) in \( Q^k_2 \). Furthermore, if we write, for example, \( N^+(v_{i,j}, v_{i,j+1}), Z^-(v_{i,j}, v_{i,j}) \) or some other illegal node-pairing then we regard the path so denoted as being the empty path.

We begin with two lemmas, the first concerning paths in a row-torus \( rt(0,1) \) in which there is a faulty node, and the second concerning paths in a row-torus \( rt(0,p - 1) \) in which there are no faults. These two lemmas are used repeatedly in the proofs of the subsequent propositions, each of which deals with a specific configuration of faults relating to the base case.
Lemma 3.3.1 Let $k \geq 4$ be even and consider the row-torus $rt(0,1)$ in $Q_2^k$ where 1 node of the row-torus is faulty. If the pair of distinct, healthy nodes $\{s, e\}$ of the row-torus is odd (resp. even) then there is a path $p(s, e)$ in the row-torus joining $s$ and $e$ of length at least $2k - 3$ (resp. $2k - 4$).

Proof: By the symmetric properties of the row-torus $rt(0,1)$, w.l.o.g. we may assume that the fault is the node $v_{0,0}$.

Suppose that $s$ and $e$ are both odd. W.l.o.g. there are four cases. (Throughout, we proceed by a case-by-case analysis, eliminating some cases by applying automorphisms of $rt(0,1)$ such as "reflections in the vertical bisecting plane" or "toroidal rotations").

Case (a) $s$ and $e$ both lie on row 0 with $s = v_{0,i}, e = v_{0,j}$ and $i < j$. Consider the path

$$ C_{j-1}^+(v_{0,i}, v_{1,i}), Z^-(v_{1,i}, v_{1,1}), (v_{1,1}, v_{1,0}), (v_{1,0}, v_{1,k-1}), N^-(v_{1,k-1}, v_{1,j}), (v_{1,j}, v_{0,j}). $$

This path has length $2k - 2$ and is as depicted in 3.1(a).

Case (b) $s$ and $e$ lie on different rows with $s = v_{0,i}, e = v_{1,j}$ and $i < j$. Consider the path

$$ C_{j-1}(v_{0,i}, v_{1,i}), Z^-(v_{1,i}, v_{1,1}), (v_{1,1}, v_{1,0}), (v_{1,0}, v_{1,k-1}), N^-(v_{1,k-1}, v_{1,j+1}), (v_{1,j+1}, v_{0,j+1}), (v_{0,j+1}, v_{0,j}), (v_{0,j}, v_{1,j}). $$

This path has length $2k - 2$ and is as depicted in Fig. 3.1(b).

Case (c) $s$ and $e$ lie on different rows with $s = v_{0,i}$ and $e = v_{1,0}$. Consider the path

$$ C_{k-1}(v_{0,i}, v_{1,i}), Z^-(v_{1,i}, v_{1,1}), (v_{1,1}, v_{1,0}). $$

This path has length $2k - 2$ and is as depicted in Fig. 3.1(c).

Case (d) $s$ and $e$ both lie on row 1 with $s = v_{1,i}, e = v_{1,j}$ and $i < j$. Consider the path

$$ N^-(v_{1,i}, v_{1,0}), (v_{1,0}, v_{1,k-1}), N^-(v_{1,k-1}, v_{1,j+1}), (v_{1,j+1}, v_{0,j+1}), (v_{0,j+1}, v_{0,j}), C_{i+1}^-(v_{0,j}, v_{1,j}). $$
3.3. The base case

This path has length $2k - 2$ and is as depicted in Fig. 3.1(d).

Suppose now that $s$ and $e$ are both even. W.l.o.g. there are three cases.

Case (e) $s$ and $e$ both lie on row 0 with $s = v_{0,i}$, $e = v_{0,j}$ and $i < j$. Consider the path

$$C^+_{j-1}(v_{0,i}, v_{1,i}), Z^-(v_{1,i}, v_{1,2}), (v_{1,2}, v_{1,1}), (v_{1,1}, v_{1,0}), (v_{1,0}, v_{1,k-1}),$$

$$N^-(v_{1,k-1}, v_{1,j+1}), (v_{1,j+1}, v_{0,j+1}), (v_{0,j+1}, v_{0,j}).$$

This path has length $2k - 4$ and is similar to the path depicted in Fig. 3.1(a).

Case (f) $s$ and $e$ lie on different rows with $s = v_{0,i}$, $e = v_{1,j}$ and $i < j$. Consider the path

$$C^+_{j-1}(v_{0,i}, v_{1,i}), Z^-(v_{1,i}, v_{1,2}), (v_{1,2}, v_{1,1}), (v_{1,1}, v_{1,0}), (v_{1,0},$$

$$v_{1,k-1}), N^-(v_{1,k-1}, v_{1,j}).$$

This path has length $2k - 4$ and is similar to the path depicted in Fig. 3.1(b).

Case (g) $s$ and $e$ both lie on row 1 with $s = v_{1,i}$, $e = v_{1,j}$ and $i < j$. Consider the path

$$N^-(v_{1,i}, v_{1,1}), (v_{1,1}, v_{1,0}), (v_{1,0}, v_{1,k-1}), N^-(v_{1,k-1}, v_{1,j+2}), (v_{1,j+2}, v_{0,j+2}),$$

$$(v_{0,j+2}, v_{0,j+1}), (v_{0,j+1}, v_{0,j}), C^-_{i+1}(v_{0,j}, v_{1,j}).$$

This path has length $2k - 4$ and is similar to the path depicted in Fig. 3.1(d).
3.3. The base case

Suppose now that one of \( s \) and \( e \) is odd and one is even, and, further, that \( s \) and \( e \) lie on the same row. W.l.o.g. there are three cases.

**Case (h)** \( s \) and \( e \) both lie on row 0 with \( s = v_{0,i} \) odd, \( e = v_{0,j} \) even and \( i < j \).

Consider the path

\[
C_{j-1}^+(v_{0,i}, v_{1,i}), Z^-(v_{1,i}, v_{1,1}), (v_{1,1}, v_{1,0}), (v_{1,0}, v_{1,k-1}), N^-(v_{1,k-1}, v_{1,j+1}), (v_{1,j+1}, v_{0,j+1}), (v_{0,j+1}, v_{0,j}).
\]

This path has length \( 2k - 3 \) and is as depicted in Fig. 3.2(h).

**Case (i)** \( s \) and \( e \) both lie on row 1 with \( s = v_{1,i} \) odd, \( e = v_{1,j} \) even and \( 0 < i < j \).

Consider the path

\[
C_{j-1}^+(v_{1,i}, v_{0,i}), Z^-(v_{0,i}, v_{0,2}), (v_{0,2}, v_{0,1}), (v_{0,1}, v_{1,1}), (v_{1,1}, v_{1,0}), (v_{1,0}, v_{1,k-1}), N^-(v_{1,k-1}, v_{1,j}).
\]

This path has length \( 2k - 3 \) and is as depicted in Fig. 3.2(i).

**Case (j)** \( s \) and \( e \) both lie on row 1 with \( s = v_{1,0} \) and \( e = v_{1,j} \) even. Consider the path

\[
(v_{1,0}, v_{1,k-1}), N^-(v_{1,k-1}, v_{1,j+2}), (v_{1,j+2}, v_{0,j+2}), (v_{0,j+2}, v_{0,j+1}), (v_{0,j+1}, v_{0,j}), C^+_1(v_{0,j}, v_{1,j}).
\]

This path has length \( 2k - 3 \) and is as depicted in Fig. 3.2(j).

Suppose now that one of \( s \) and \( e \) is odd and one is even, and, further, that \( s \) and \( e \) lie on different rows. W.l.o.g. there are five cases.
3.3. The base case

Case (k) $s$ lies on row 0 and $e$ lies on row 1 with $s = v_{0i}$ odd, $e = v_{1j}$ even and $i < j$. Consider the path

$$C^+_{j-1}(v_{0i}, v_{1i}), Z^-(v_{1i}, v_{11}), (v_{11}, v_{10}), (v_{10}, v_{1,k-1}), N^-(v_{1,k-1}, v_{1j}).$$

This path has length $2k - 3$ and is as depicted in Fig. 3.3(k).

Case (l) $s$ and $e$ lie on different rows with $s = v_{0i}$ odd, $e = v_{1i}$ even and $i \neq 1$. Consider the path

$$Z^-(v_{0i}, v_{03}), (v_{03}, v_{02}), (v_{02}, v_{12}), (v_{12}, v_{11}), (v_{11}, v_{10}), (v_{10}, v_{1,k-1}), N^-(v_{1,k-1}, v_{1j}).$$

This path has length $2k - 3$ and is as depicted in Fig. 3.3(l).

Case (m) $s$ and $e$ lie on different rows with $s = v_{0i}$ even, $e = v_{1i}$ odd and $i < j$. Consider the path

$$C^+_{j-1}(v_{0i}, v_{1i}), Z^-(v_{1i}, v_{12}), (v_{12}, v_{11}), (v_{11}, v_{10}), (v_{10}, v_{1,k-1}), N^-(v_{1,k-1}, v_{1,j+1}), (v_{1,j+1}, v_{0,j+1}), (v_{0,j+1}, v_{0,j}), (v_{0,j}, v_{1j}).$$

This path has length $2k - 3$ and is as depicted in Fig. 3.3(m).
3.3. The base case

Case (n) $s$ and $e$ lie on different rows with $s = v_{0,i}$ even and $e = v_{1,0}$. Consider the path

$$C^+_{j-1}(v_{0,i}, v_{1,i}), Z^-(v_{1,i}, v_{1,2}), (v_{1,2}, v_{1,1}), (v_{1,1}, v_{1,0}).$$

This path has length $2k - 3$ and is as depicted in Fig. 3.3(n).

Case (o) $s$ and $e$ lie on different rows with $s = v_{0,i}$ even, $e = v_{1,i}$ odd. Consider the path

$$Z^-(v_{0,i}, v_{0,2}), (v_{0,2}, v_{0,1}), (v_{0,1}, v_{1,1}), (v_{1,1}, v_{1,0}), (v_{1,0},$$
$$v_{1,k-1}), N^-(v_{1,k-1}, v_{1,j+1}), (v_{1,j+1}, v_{1,j}).$$

This path has length $2k - 3$ and is as depicted in Fig. 3.3(o).

The result follows. □

The following lemma proves to be useful throughout.

Lemma 3.3.2 Let $k \geq 4$ be even and consider the row-torus $rt(0, p-1)$ in $Q^p_2$ where $2 \leq p \leq k$. If the pair of distinct nodes $\{s, e\}$ of the row-torus is odd (resp. even) then there is a path $\rho(s, e)$ in the row-torus joining $s$ and $e$ of length $pk - 1$ (resp. $pk - 2$).

Proof: We proceed by induction on $p$. Suppose that $p = 2$ and consider the row-torus $rt(0, 1)$. W.l.o.g. we may assume that $e = v_{0,0}$.

Suppose that $s = v_{0,i}$ is odd. The path

$$C^+_{k-1}(s, v_{1,i}), Z^-(v_{1,i}, v_{1,1}), (v_{1,1}, v_{1,0}), (v_{1,0}, e)$$

has length $2k - 1$.

Suppose that $s = v_{0,i}$ is even. The path

$$C^+_{k-1}(s, v_{1,i}), Z^-(v_{1,i-2}, v_{1,2}), (v_{1,2}, v_{1,1}), (v_{1,1}, v_{1,0}), (v_{1,0}, e)$$

has length $2k - 2$.

Suppose that $s = v_{1,i}$ is odd. The path

$$C^+_{k-1}(s, v_{0,i}), Z^-(v_{0,i}, e)$$

has length $2k - 1$. 

3.3. The base case

Suppose that \( s = v_{1,i} \) is even. The path

\[ C_{k-1}^+(s, v_{0,1}), Z^-(v_{0,i}, v_{0,1}), (v_{0,1}, e) \]

has length \( 2k - 2 \). So the result holds for \( p = 2 \).

Suppose, as our induction hypothesis, that the result holds for all \( p \) such that \( 1 \leq p < q \), where \( 1 < q \leq k - 1 \). Consider \( rt(0, q) \).

Case (a) It is not the case that \( s \) lies on row 0 and \( e \) lies on row \( q \), and it is not the case that \( s \) lies on row \( q \) and \( e \) lies on row 0.

W.l.o.g. assume that \( s \) and \( e \) lie in \( rt(0, q - 1) \). By the induction hypothesis, there is a path \( \rho(s, e) \) in \( rt(0, q - 1) \) of length \( qk - 1 \) (resp. \( qk - 2 \)) if \( \{s, e\} \) is odd (resp. even). For a node \( r \) in row \( q - 1 \), if it is not linked with its neighbor on the same row, then either \( r = s \), or \( r = e \), or it is not on path \( \rho(s, e) \). As there is at most one node not in path \( \rho(s, e) \), there are at least \( \lceil \frac{k-1}{2} \rceil \) edges on row \( q - 1 \) which are also on the path \( \rho(s, e) \). Hence, the path \( \rho(s, e) \) must contain a link \( (v_{q-1,i}, v_{q-1,i+1}) \)
lying on row \( q - 1 \).

Consider the path

\[ \rho(s, v_{q-1,i}), (v_{q-1,i}, v_{q,i}), (v_{q,i}, v_{q,i-1}), (v_{q,i-1}, v_{q,i-2}), \ldots, (v_{q,i+2}, v_{q,i+1}), (v_{q,i+1}, v_{q-1,i+1}), \rho(v_{q-1,i+1}, e). \]

This path is as required (with reference to our construction as detailed at the beginning of this section, an alternative description of this path would be as that obtained by joining \( \rho(s, e) \) to the cycle

\[ (v_{q,0}, v_{q,1}, v_{q,2}), \ldots, (v_{q,k-2}, v_{q,k-1}), (v_{q,k-1}, v_{q,0}) \]

over the links \( (v_{q-1,i}, v_{q-1,i+1}) \) and \( (v_{q,i}, v_{q,i+1}) \).

Case (b) The node \( s \) lies on row 0 and the node \( e \) lies on row \( q \).

If \( e = v_{q,i} \), then define \( e' = v_{q-1,i-1} \). Note that \( e \) is odd if, and only if, \( e' \) is odd.

By the induction hypothesis, there is a path \( \rho(s, e') \) in \( rt(0, q - 1) \) of length \( qk - 1 \) (resp. \( qk - 2 \)) if \( \{s, e\} \) is odd (resp. even). The path

\[ \rho(s, e'), (e', v_{q,i-1}), (v_{q,i-1}, v_{q,i-2}), (v_{q,i-2}, v_{q,i-3}), \ldots, (v_{q,i+1}, e) \]
3.3. The base case

is as required.

The result follows by induction. \(\square\)

We now deal with first scenario in the base case.

**Proposition 3.3.1** Consider the k-ary 2-cube \(Q^k_2\) where \(k \geq 6\) is even and where 2 of the nodes are faulty. Let \(s\) and \(e\) be any two distinct, non-faulty nodes. There is a path of length at least \(k^2 - 5\) (resp. \(k^2 - 6\)) from \(s\) to \(e\) if \(\{s, e\}\) is odd (resp. even).

**Proof:** W.l.o.g. suppose that the two faulty nodes are \(f_0 = v_{0,0}\) and \(f_1 = v_{p,q}\) with \(p \neq 0\). We begin by partitioning \(Q^k_2\) into 3 or 4 row-tori. If \(p \in \{1, 2, k-2, k-1\}\) then:

- if \(p = 1\) or \(p = 2\) then we partition \(Q^k_2\) into \(A = rt(k-1,0), B = rt(1,2)\) and \(X = rt(3,k-2)\);

- if \(p = k-2\) or \(p = k-1\) then we partition \(Q^k_2\) into \(A = rt(0,1), X = rt(2,k-3)\) and \(B = rt(k-2,k-1)\).

If \(p \notin \{1, 2, k-2, k-1\}\) then:

- if \(p \neq 3\) is odd then we partition \(Q^k_2\) into \(A = rt(0,1), X = rt(2,p-2), B = rt(p-1,p)\) and \(Y = rt(p+1,k-1)\);

- if \(p = 3\) then we partition \(Q^k_2\) into \(A = rt(k-1,0), X = rt(1,2), B = rt(3,4)\) and \(Y = rt(5,k-2)\);

- if \(p\) is even then we partition \(Q^k_2\) into \(A = rt(0,1), X = rt(2,p-1), B = rt(p,p+1)\) and \(Y = rt(p+2,k-1)\).

The outcome is that we have one of the two partitioned structures as in Fig.3.4, where consecutive row-tori are joined by column links. In particular, w.l.o.g. we may assume that: when the partition involves 3 row-tori, we have the situation as in Fig. 3.4(a), with \(f_0 = v_{0,0} \in A = rt(0,1), X = rt(2,k-3)\) and \(f_1 \in B = rt(k-2,k-1)\); and when the partition involves 4 row-tori, we have the situation as in Fig. 3.4(b), with \(f_0 = v_{0,0} \in A = rt(0,1), X = rt(2,q-1), f_1 \in B = rt(q,q+1)\) and \(Y = rt(q+2,k-1)\), for some even \(q\) where \(4 \leq q \leq k - 4\).
Throughout the proof, \( \epsilon = 1 \) if \( \{s, e\} \) is odd, and \( \epsilon = 2 \) if \( \{s, e\} \) is even.

Case (a) \( Q^k_2 \) is partitioned into 3 row-tori.

Sub-case (i) The nodes \( s \) and \( e \) both lie in \( A \).

By Lemma 3.3.1, there exists a path \( \rho_A(s, e) \) in \( A \) of length at least \( 2k - 2 - \epsilon \). A simple counting argument yields that there is at least one link of \( \rho_A(s, e) \) lying on row 1; w.l.o.g. let \( (v_{1,i}, v_{1,i+1}) \) be such a link (the case when the link is \( (v_{1,i+1}, v_{1,i}) \) is almost identical). By Lemma 3.3.2, there exists a path \( \rho_X(v_{2,i}, v_{2,i+1}) \) in \( X \) of length \( k(k - 4) - 1 \). Let \( \rho(s, e) \) be obtained by joining \( \rho_A(s, e) \) to \( \rho_X(v_{2,i}, v_{2,i+1}) \) over \( (v_{1,i}, v_{1,i+1}) \). Again, a simple counting argument yields that there are at least two non-incident links of \( \rho(s, e) \) lying on row \( k - 3 \); w.l.o.g. let \( (v_{k-3,j}, v_{k-3,j+1}) \) be such a link where \( v_{k-2,j} \neq f_1 \neq v_{k-2,j+1} \). By Lemma 3.3.1, there exists a path \( \rho_B(v_{k-2,j}, v_{k-2,j+1}) \) in \( B \) of length at least \( 2k - 3 \). The path obtained by joining \( \rho(s, e) \) to \( \rho_B(v_{k-2,j}, v_{k-2,j+1}) \) over \( (v_{k-3,j}, v_{k-3,j+1}) \) has length at least \( k^2 - 4 - \epsilon \).

Sub-case (ii) The node \( s \) is in \( A \) and the node \( e \) is in \( X \).

Choose \( v_{1,i} \) such that \( v_{1,i} \) is odd if, and only if, \( s \) is even, and \( v_{2,i} \neq e \) (such a node \( v_{1,i} \) exists due to there are at least \( \frac{k}{2} \geq 2 \) nodes on row 1 that have different parity from node \( e \)). By Lemma 3.3.1, there exists a path \( \rho_A(s, v_{1,i}) \) in \( A \) of length at least \( 2k - 3 \). By Lemma 3.3.2, there exists a path \( \rho_X(v_{2,i}, e) \) in \( X \) of length \( k(k - 4) - \epsilon \).
Let \( \rho(s,e) \) be the path

\[
\rho_A(s,v_{1,i}),(v_{1,i},v_{2,i}),\rho_X(v_{2,i},e).
\]

A simple counting argument yields that \( \rho(s,e) \) contains at least two non-incident links on row \( k-3 \); w.l.o.g. let \((v_{k-3,j},v_{k-3,j+1})\) be a link of \( \rho(s,e) \) such that \( v_{k-2,j} \neq f_1 \neq v_{k-2,j+1} \). By Lemma 3.3.1, there exists a path \( \rho_B(v_{k-2,j},v_{k-2,j+1}) \) in \( B \) of length at least \( 2k - 3 \). The path obtained by joining \( \rho(s,e) \) to \( \rho_B(v_{k-2,j},v_{k-2,j+1}) \) over \((v_{k-3,j},v_{k-3,j+1})\) has length at least \( k^2 - 4 - \epsilon \).

Sub-case (iii) The node \( s \) is in \( A \) and the node \( e \) is in \( B \).

Choose \( v_{1,i} \) such that \( v_{1,i} \) is odd if, and only if, \( s \) is even, and \( v_{1,i} \neq s \). By Lemma 3.3.1, there exists a path \( \rho_A(s,v_{1,i}) \) in \( A \) of length at least \( 2k - 3 \). Choose \( v_{k-2,j} \) such that \( v_{k-2,j} \) is odd if, and only if, \( e \) is even, and \( f_1 \neq v_{k-2,j} \). By Lemma 3.3.1, there exists a path \( \rho_B(v_{k-2,j},e) \) in \( B \) of length at least \( 2k - 3 \). By Lemma 3.3.2, there exists a path \( \rho_X(v_{2,i},v_{k-3,j}) \) in \( X \) of length \( k(k - 4) - \epsilon \). The path

\[
\rho_A(s,v_{1,i}),(v_{1,i},v_{2,i}),\rho_X(v_{2,i},v_{k-3,j}),(v_{k-3,j},v_{k-2,j}),\rho_B(v_{k-2,j},e)
\]

has length at least \( k^2 - 4 - \epsilon \).

Sub-case (iv) The nodes \( s \) and \( e \) both lie in \( X \).

By Lemma 3.3.2, there exists a path \( \rho_X(s,e) \) in \( X \) of length \( k(k - 4) - \epsilon \). A simple counting argument yields that \( \rho_X(s,e) \) always contains at least one link on row 2 and also that there are two non-incident links on row \( k - 3 \), unless we have the special situation where \( k = 6, s \) and \( e \) have a common neighbour on row \( k - 3 \) with this neighbour not lying on \( \rho_X(s,e) \), and neither \( s \) nor \( e \) is adjacent on \( \rho_X(s,e) \) to a node on row \( k - 3 \). Suppose that there are two non-incident links on row \( k - 3 \). W.l.o.g. let \((v_{k-3,j},v_{k-3,j+1})\) and \((v_{2,i},v_{2,i+1})\) be links of \( \rho_X(s,e) \) where \( v_{k-2,j} \neq f_1 \neq v_{k-2,j+1} \).

By Lemma 3.3.1, there exists a path \( \rho_B(v_{k-2,j},v_{k-2,j+1}) \) (resp. \( \rho_A(v_{1,i},v_{1,i+1}) \)) in \( B \) (resp. \( A \)) of length at least \( 2k - 3 \). W.l.o.g. suppose that the nodes \( v_{k-3,j}, v_{k-3,j+1}, v_{2,i} \) and \( v_{2,i+1} \) come in that order as we move along the path \( \rho_X(s,e) \). The path

\[
\rho_X(s,v_{k-3,j}),(v_{k-3,j},v_{k-2,j}),\rho_B(v_{k-2,j},v_{k-2,j+1}),(v_{k-2,j+1},v_{k-3,j+1}),
\rho_X(v_{k-3,j+1},v_{2,i}),(v_{2,i},v_{1,i}),\rho_A(v_{1,i},v_{1,i+1}),
\]

has length at least \( k^2 - 4 - \epsilon \).
3.3. The base case

\[(v_{1,i+1}, v_{2,i+1}), \rho_X(v_{2,i+1}, e)\]

has length at least \(k^2 - 4 - \epsilon\).

Alternatively, suppose that we are in the special situation described above (and so \(k = 6\)). W.l.o.g. suppose that \(s = v_{3,0}\) and \(e = v_{3,2}\); so, the path \((v_{3,3}, v_{3,4}), (v_{3,4}, v_{3,5})\) is a sub-path of \(\rho_X(s, e)\). If \(f_1 \neq v_{4,4}\) then we can find two links \((v_{3,j}, v_{3,j+1})\) and \((v_{2,i}, v_{2,i+1})\) of \(\rho_X(s, e)\), as above, and so obtain our path as required. So, suppose that \(f_1 = v_{4,4}\). Let \(\rho_B(v_{4,3}, v_{4,5})\) be the path

\[(v_{4,3}, v_{4,2}), (v_{4,2}, v_{4,1}), (v_{4,1}, v_{4,0}), (v_{4,0}, v_{4,5})\]

and join \(\rho_X(s, e)\) to \(\rho_B(v_{4,3}, v_{4,5})\) over \((v_{3,3}, v_{3,4}), (v_{3,4}, v_{3,5})\) to obtain the path \(\rho(s, e)\) of length \(16 - \epsilon\). We can now join \(\rho(s, e)\) to the cycle induced by the nodes on row 5, over two appropriate links, and to an appropriate path \(\rho_A(v_{1,i}, v_{1,i+1})\) in \(A\) of length at least 9, as we did above, to obtain our required path of length at least \(32 - \epsilon\) (that is, \(k^2 - 4 - \epsilon\)).

The remaining sub-cases are essentially identical to those already considered.

Case (b) \(Q_2^5\) is partitioned into 4 row-tori.

If \(s\) and \(e\) lie in \(A \cup X \cup B\) then by the analysis for Case (a), there is a path \(\rho(s, e)\) in \(A \cup X \cup B\) (and the connecting column links) of length at least \(k(q + 2) - 4 - \epsilon\) (note that all paths constructed in Case (a) actually lie in the row-torus induced by \(A \cup X \cup B\)). A simple counting argument yields that there is at least one link of \(\rho(s, e)\) on row \(q + 1\) or on row 0; w.l.o.g. suppose that it is row \(q + 1\) and let \((v_{q+1,j}, v_{q+1,j+1})\) be such a link. By Lemma 3.3.2, there exists a path \(\rho_Y(v_{q+2,j}, v_{q+2,j+1})\) in \(Y\) of length \(k(k - 1 - q - 1) - 1\). Join \(\rho(s, e)\) to \(\rho_Y(v_{q+2,j}, v_{q+2,j+1})\) over \((v_{q+1,j}, v_{q+1,j+1})\) to obtain a path of length at least \(k^2 - 4 - \epsilon\). A similar argument holds should \(s\) and \(e\) lie in \(B \cup Y \cup A\).

Necessarily, the only remaining case is when \(s\) lies in \(X\) and \(e\) lies in \(Y\). Let \(v_{0,i}\) be such that \(s\) and \(e\) do not lie on column \(i\) and \(v_{0,i}\) is odd if, and only if, \(e\) is odd. By Lemma 3.3.2, there exists a path \(\rho_Y(v_{k-1,i}, e)\) in \(Y\) of length \(k(k - 1 - q - 1) - 1\). Let \(v_{1,j}\) be such that \(s\) does not lie on column \(j\) and \(v_{1,j}\) is odd if, and only if, \(s\) is odd. By Lemma 3.3.2, there exists a path \(\rho_X(s, v_{2,j})\) in \(X\) of length \(k(q - 2) - 1\).
By Lemma 3.3.1, there exists a path $\rho_A(v_{1,j},v_{0,i})$ in $A$ of length at least $2k - 2 - \epsilon$. Let $\rho(s,e)$ be the path

$$\rho_X(s,v_{2,j}), (v_{2,j},v_{1,j}), \rho_A(v_{1,j},v_{0,i}), (v_{0,i},v_{k-1,i}), \rho_Y(v_{k-1,i},e).$$

Necessarily, there are at least two non-incident links of $\rho_X(s,v_{2,j})$ on row $q-1$; w.l.o.g. let $(v_{q-1,m},v_{q-1,m+1})$ be such a link with $v_{q,m} \neq f_i = v_{q,m+1}$. By Lemma 3.3.1, there exists a path $\rho_B(v_{q,m},v_{q,m+1})$ in $B$ of length $2k - 3$. The path obtained by joining $\rho(s,e)$ to $\rho_B(v_{q,m},v_{q,m+1})$ over $(v_{q-1,m},v_{q-1,m+1})$ has length at least $k^2 - 4 - \epsilon$. The result follows.

We deal with the case when $k = 4$ later (as we do also for subsequent propositions).

The next proposition deals with the next scenario in the base case.

**Proposition 3.3.2** Consider the $k$-ary 2-cube $Q_k^2$ where $k \geq 6$ is even and where 1 of the nodes is faulty. Let $s$ and $e$ be any two distinct, non-faulty nodes. There is a path of length at least $k^2 - 3$ (resp. $k^2 - 4$) from $s$ to $e$ if $\{s, e\}$ is odd (resp. even).

**Proof:** The proof is a much simplified version of the proof of Proposition 3.3.1. Essentially, we partition $Q_k^2$ into 2 row-tori, $A = rt(0,1)$ and $X = rt(2,k-1)$, and follow the constructions in Sub-cases (a.i), (a.ii) and (a.iv). The result follows. □

We now consider when there are only faulty links in $Q_k^2$, but first we construct some basic hamiltonian circuits on row-tori. Consider the row-torus $rt(0,p-1)$ in $Q_k^2$, for some even $p$ where $2 \leq p \leq k-1$. For every even $i \in \{0,1,\ldots, p-2\}$, build the following cycle $C_i$:

$$(v_{i,0},v_{i,1}), (v_{i,1},v_{i,2}), \ldots, (v_{i,k-2},v_{i,k-1}), (v_{i,k-1},v_{i+1,k-1}),$$

$$(v_{i+1,k-1},v_{i+1,k-2}), \ldots, (v_{i+1,1},v_{i+1,0}), (v_{i+1,0},v_{i,0}).$$

Join the cycle $C_0$ to the cycle $C_2$ over the links $(v_{1,0},v_{1,1})$ and $(v_{2,0},v_{2,1})$, and denote the resulting cycle by $E_{0,0}$. Now join $E_{0,0}$ to the cycle $C_4$ over the links $(v_{3,0},v_{3,1})$ and $(v_{4,0},v_{4,1})$, and denote the resulting cycle by $E_{0,0}$ also. Proceed in this way to obtain the hamiltonian cycle $E_{0,0}$ of the row-torus $rt(0,p-1)$ rooted at $v_{0,0}$.

If $3 \leq p \leq k-1$ is odd then build the cycle $E_{0,0}$ in the row-torus $rt(0,p-2)$ and join it to the cycle induced by the nodes on row $p-1$, over the links $(v_{p-2,0},v_{p-2,1})$...
3.3. The base case

Figure 3.5: The hamiltonian cycle $E_{0,0}$ in $rt(0, 6)$ in $Q_2^7$.

and $(v_{p-1,0}, v_{p-1,1})$; denote the resulting cycle as the cycle $E_{0,0}$ of $rt(0, p - 1)$ rooted at $v_{0,0}$. The hamiltonian cycle $E_{0,0}$ in $rt(0, 6)$ in $Q_2^7$ can be visualised as in Fig.3.5.

Note that we also have the hamiltonian cycles $E_{0,i}$ of $rt(0, p - 1)$, for all $p \in \{2, 3, \ldots, k\}$ and $i \in \{1, 2, \ldots, k - 1\}$, obtained by starting the above process at the root-node $v_{0,i}$ as opposed to node $v_{0,0}$.

**Proposition 3.3.3** Consider the $k$-ary 2-cube $Q_2^k$ where $k \geq 6$ is even and where there is 1 faulty link. Let $s$ and $e$ be any two distinct nodes in the row-torus $rt(0, p - 1)$, where $2 \leq p \leq k$. There is a path in $rt(0, p - 1)$ from $s$ to $e$ of length $pk - 1$ (resp. $pk - 2$) if $\{s, e\}$ is odd (resp. even).

**Proof:** By Lemma 3.3.2, we may assume that the faulty link lies in $rt(0, p - 1)$. W.l.o.g. we may assume that the faulty link is either $(v_{a,0}, v_{a+1,0})$ or $(v_{a,0}, v_{a,1})$, where $0 \leq a \leq p - 2$. As before, $\epsilon = 1$ if $\{s, e\}$ is odd, and $\epsilon = 2$ if $\{s, e\}$ is even.

Case (a) $a = 0$, and the faulty link is $(v_{0,0}, v_{1,0})$.

Sub-case (i) $s$ and $e$ lie on row 0.

If $s = v_{0,i}$ and $e = v_{0,j}$ then w.l.o.g. we may assume that $i < j$ and that it is not the case that both $i = 0$ and $j = k - 1$.

Suppose that it is not the case that $i = 1$ and $j = k - 1$. Let $\rho_0(s, e)$ be the path

$$(s, v_{0,i-1}), (v_{0,i-1}, v_{0,i-2}), \ldots, (v_{0,j+1}, e).$$
3.3. The base case

Figure 3.6: Joining $\rho_0(s, e)$ to the amended cycle $C$.

Note that the length of $\rho_0(s, e)$ is odd if, and only if, $\{s, e\}$ is odd; so, there are an even number of nodes on row 0 that are not on $\rho_0(s, e)$ if, and only if, $\{s, e\}$ is odd. Let $C$ be the cycle induced by the nodes on row 1. Iteratively join $C$ to appropriate links $(v_{0,i}, v_{0,i+1})$ over $(v_{1,i}, v_{1,i+1})$ so that the nodes used on row 0 do not already appear on $\rho_0(s, e)$. Links should be replaced (by paths) so that if $\{s, e\}$ is odd (resp. even) then every node of $rt(0, 1)$ appears on the (amended) cycle $C$ or on $\rho_0(s, e)$ (resp. except one). Join $\rho_0(s, e)$ to $C$ over two corresponding links (this is always possible) and denote the new path by $\rho_A(s, e)$. The path $\rho_A(s, e)$ has length $2k - \varepsilon$.

This construction can be visualised in Fig.3.6, where the dashed links show how $\rho_0(s, e)$ is joined to the amended $C$.

Suppose that $i = 1$ and $j = k - 1$. Let $\rho_0(s, e)$ be the path

$$(s, v_{0,2}), (v_{0,2}, v_{0,3}), \ldots, (v_{0,k-2}, e).$$

Let $C$ be the cycle induced by the nodes on row 1. Join $\rho_0(s, e)$ to $C$ over $(v_{0,1}, v_{0,2})$ and $(v_{1,1}, v_{1,2})$, and denote the new path by $\rho_A(s, e)$. The path $\rho_A(s, e)$ has length $2k - 2$.

If $p = 2$ then we are done. If $p > 3$ then let $D$ be the hamiltonian cycle $E_{2,0}$ in the row-torus $rt(2, p-1)$, and if $p = 3$ then let $D$ be the cycle induced by the nodes on row 2. Join $\rho_A(s, e)$ to $D$ over two corresponding links, and the resulting path is as required.

Sub-case (ii) $s$ lies on row 0 and $e$ lies on row 1.

Let $s = v_{0,i}$ and $e = v_{1,j}$; w.l.o.g. we may assume that $i \neq k - 1$. If $i \neq 1$ then let $e'$ be a neighbour of $s$ on row 0 that does not lie in the same column as $e$. If $i = 1$ and $j \neq 2$ then let $e' = v_{0,2}$. Either way, let $\rho_0(s, e')$ be a path on row 0 of length $k - 1$. If $i = 1$ and $j = 2$ then let $e' = v_{0,3}$ and let $\rho_0(s, e')$ be a path on row 0 of length $k - 2$. 
Let \( s' \) be the neighbour of \( e' \) on row 1 and let \( \rho_1(s', e) \) be a path on row 0 which contains the link \((v_{1,0}, v_{1,1})\). Define the path \( \rho_A(s, e) \) as
\[
\rho_0(s, e'), (e', s'), \rho_1(s', e).
\]

Iteratively join \( \rho_A(s, e) \) to appropriate links \((v_{1,1}, v_{1,i+1})\) over \((v_{0,i}, v_{0,i+1})\) so that the nodes used on row 1 do not already appear on \( \rho_A(s, e) \). Links should be replaced (by paths) so that if \( \{s, e\} \) is odd (resp. even) then every node of \( rt(0, 1) \) appears on (the amended) \( \rho_A(s, e) \) (resp. except one).

If \( p = 2 \) then we are done. If \( p > 3 \) then let \( D \) be the hamiltonian cycle \( E_{2,0} \) in the row-torus \( rt(2, p-1) \), and if \( p = 3 \) then let \( D \) be the cycle induced by the nodes on row 2. Join \( \rho_A(s, e) \) to \( D \) over the links \((v_{1,0}, v_{1,1})\) and \((v_{2,0}, v_{2,1})\). The resulting path is as required.

Note that if \( p = 2 \) then we have covered all cases, so henceforth we assume that \( p \geq 3 \).

Sub-case (iii) \( s \) lies on row 0 and \( e \) lies on rows 2, 3, \ldots, \( p - 1 \).

Suppose that \( s = v_{0,i} \). If \( i \neq 1 \) then define \( e' = v_{0,i-1} \), and if \( i = 1 \) then define \( e' = v_{0,i+1} \). Define the path \( \rho_0(s, e') \) to be the path on row 0 of length \( k - 1 \). Let \( e'' \) be the neighbour of \( e' \) on row 1, and let \( e''' \) be a neighbour of \( e'' \) on row 1 that does not lie in the same column as \( e \). Define the path \( \rho_1(e'', e''') \) as the path of length \( k - 1 \) on row 1. Define the path \( \rho_A(s, e''') \) as
\[
\rho_0(s, e'), (e', e''), \rho_1(e'', e''').
\]
The path \( \rho_A(s, e''') \) has length \( 2k - 1 \).

Let \( s' \) be the neighbour of \( e''' \) on row 2. If \( p \geq 4 \) then by Lemma 3.3.2, there is a path \( \rho_X(s', e) \) in \( rt(2, p - 1) \) of length \( k(p - 2) - \epsilon \), and the path
\[
\rho_A(s, e'''), (e''', s'), \rho_X(s', e)
\]
is as required. If \( p = 3 \) then define the path \( \rho_X(s', e) \) to be a path on row 2, and let \( \rho(s, e) \) be the path
\[
\rho_A(s, e'''), (e''', s'), \rho_X(s', e).
\]
3.3. The base case

Iteratively join $\rho(s, e)$ to appropriate links $(v_{2,i}, v_{2,i+1})$ over $(v_{1,i}, v_{1,i+1})$ so that the nodes used on row 2 do not already appear on $\rho(s, e)$. Links should be replaced (by paths) so that if $\{s, e\}$ is odd (resp. even) then every node of row 2 appears on the amended path (resp. except one). The resulting path is as required.

Sub-case (iv) $s$ and $e$ lie on row 1.

Proceed as in Sub-case (i) to build a path (analogous to) $\rho_A(s, e)$. The path $\rho_A(s, e)$ is such that it contains a link on row 1. Join $\rho_A(s, e)$ to the cycle $D$, as constructed in Sub-case (i) and over corresponding links, to obtain a required path.

Sub-case (v) $s$ lies on rows $1, 2, \ldots, p - 1$ and $e$ lies on rows $2, 3, \ldots, p - 1$.

By Lemma 3.3.2, there exists a path $\rho(s, e)$ in $rt(1, p - 1)$ of length $(p - 1)k - \epsilon$. There is at least one link of $\rho(s, e)$ on row 1 that is not incident with $v_{1,0}$. Join $\rho(s, e)$ to the cycle induced by the nodes on row 0 over two corresponding links to obtain a required path.

Case (b) $0 \neq a \neq p - 2$ and the faulty link is $(v_{a,0}, v_{a+1,0})$.

Sub-case (i) $s$ and $e$ lie on rows $0, 1, \ldots, a$.

By Lemma 3.3.2, there is a path $\rho_A(s, e)$ in $rt(0, a)$ of length $(a + 1)k - \epsilon$. Either: there exist 2 disjoint links of $\rho_A(s, e)$ on row $a$, and so we have a link of $\rho_A(s, e)$ on row $a$ that is not incident with $v_{a,0}$; or $k = 6$ and the nodes $v_{a,2}, v_{a,3}, v_{a,4}$ constitute $s$, $e$ and a node not on $\rho_A(s, e)$. However, in this latter case, let $E_{0,0}$ be the hamiltonian cycle in $rt(0, a)$ but with the sub-path from $s$ to $e$ involving (some of) the nodes $v_{a,2}, v_{a,3}, v_{a,4}$ removed (so, the length of this sub-path is 1, if $\{s, e\}$ is odd, and 2, if $\{s, e\}$ is even). Either way, we obtain a path, call it $\rho_A(s, e)$, in $rt(0, a)$ of length $(a + 1)k - \epsilon$ with the property that there is a link of $\rho_A(s, e)$ on row $a$ that is not incident with $v_{a,0}$.

Join $\rho_A(s, e)$ to the hamiltonian cycle $E_{a+1,0}$ of $rt(a + 1, p - 1)$, over some appropriate links, and the path obtained is as required.

Sub-case (ii) $s$ lies on rows $0, 1, \ldots, a$ and $e$ lies on rows $a + 1, a + 2, \ldots, p - 1$.

Suppose that we can choose $e'$ on row $a$ such that: $v_{a,0} \neq e' \neq s$; $e$ and $e'$ are not adjacent; and $\{s, e'\} = \{s, e\}$. If so then by Lemma 3.3.2, there is a path $\rho_A(s, e')$ in $rt(0, a)$ of length $(a + 1)k - \epsilon$ so that $e$ is not adjacent to $e'$. Define $s'$ to be
3.3. The base case

the neighbour of $e'$ on row $a + 1$. By Lemma 3.3.2, there is a path $\rho_X(s', e)$ in $rt(a + 1, p - 1)$ of length $(p - a - 1)k - 1$. The path

$$\rho_A(s, e'), (e', s'), \rho_X(s', e)$$

is as required.

Alternatively, suppose that $e'$ does not exist. This only happens when $k = 6$, and $(s = v_{a,2}$ and $e = v_{a+1,4})$ or $(s = v_{a,4}$ and $e = v_{a+1,2})$. Define $e' = v_{a,3}$ and let $E_{0,0}$ be the hamiltonian cycle in $rt(0, a)$ with the link $(s, e')$ removed; call this path $\rho_A(s, e')$. By Lemma 3.3.2, there is a path $\rho_X(v_{a+1,3}, e)$ in $rt(a + 1, p - 1)$ of length $(p - a - 1)k - 1$. The path

$$\rho_A(s, e'), (e', v_{a+1,3}), \rho_X(v_{a+1,3}, e)$$

is as required.

Case (c) $a = 0$ and the faulty link is $(v_{0,0}, v_{0,1})$.

Sub-case (i) $s$ and $e$ lie on row 0.

Let $\rho_0(s, e)$ be the path on row 0 which contains the faulty link $(v_{0,0}, v_{0,1})$, and let $C$ be the cycle induced by the nodes on row 1. Join $\rho_0(s, e)$ to $C$ over the links $(v_{0,0}, v_{0,1})$ and $(v_{1,0}, v_{1,1})$, and denote the resulting path by $\rho(s, e)$. Iteratively join $\rho(s, e)$ to appropriate links $(v_{0,i}, v_{0,i+1})$ over $(v_{1,i}, v_{1,i+1})$ so that the nodes used on row 0 do not already appear on $\rho(s, e)$. Links should be replaced (by paths) so that if $\{s, e\}$ is odd (resp. even) then every node of row 0 appears on the amended path (resp. except one). Denote the amended path by $\rho(s, e)$ also.

If $p > 3$ then let $D$ be the hamiltonian cycle $E_{2,0}$ in $rt(2, p - 1)$, and if $p = 3$ then let $D$ be the cycle induced by the nodes of row 2. Joining $\rho(s, e)$ to $D$ over two corresponding links yields a path as required.

Sub-case (ii) $s$ lies on row 0 and $e$ lies on row 1.

Suppose that $s = v_{0,i}$ and $e = v_{1,j}$. W.l.o.g. we may assume that $i$ is odd.

If $\{s, e\}$ is odd and $1 \leq j < i$ then define $\rho(s, e)$ as

$$C_0^+(s, v_{1,i}), Z^-(v_{1,i}, v_{1,j+2}), (v_{1,j+2}, v_{1,j+1}), (v_{1,j+1}, v_{0,j+1}), (v_{0,j+1}, v_{0,j}), C_1^-(v_{0,j}, e).$$
3.3. The base case

If \(\{s, e\}\) is odd and \(i < j \leq k - 1\) then define \(\rho(s, e)\) as

\[
C^-_1(s, v_{i,i}), Z^+(v_{i,i}, v_{1,j-2}), (v_{1,j-2}, v_{1,j-1}), (v_{1,j-1}, v_{0,j-1}),
(v_{0,j-1}, v_{0,j}), C^+_k(v_{0,j}, e).
\]

If \(\{s, e\}\) is odd and \(i = j\) then define \(\rho(s, e)\) as \(C^+_0(s, e)\), and if \(i \neq 1\) then define \(C\) as the cycle

\[
C^-_1(v_{0,i-1}, v_{1,i-1}), (v_{1,i-1}, v_{0,i-1}).
\]

If \(\{s, e\}\) is even and \(2 \leq j < i\) then define \(\rho(s, e)\) as

\[
C^+_0(s, v_{i,i}), Z^-(v_{i,i}, v_{1,j+3}), (v_{1,j+3}, v_{1,j+2}), (v_{1,j+2}, v_{0,j+2}),
(v_{0,j+2}, v_{0,j+1}, v_{0,j}), C^-_0(v_{0,j}, e).
\]

If \(\{s, e\}\) is even and \(j = 0\) then define \(\rho(s, e)\) as

\[
C^-_1(s, v_{i,i}), Z^+(v_{i,i}, v_{1,k-1}), (v_{1,k-1}, e).
\]

If \(\{s, e\}\) is even and \(i < j \leq k - 1\) then define \(\rho(s, e)\) as

\[
C^-_1(s, v_{i,i}), Z^+(v_{i,i}, v_{1,j-3}), (v_{1,j-3}, v_{1,j-2}), (v_{1,j-2}, v_{1,j-1}), (v_{1,j-1}, v_{0,j-1}), (v_{0,j-1}, v_{0,j}), C^+_0(v_{0,j}, e).
\]

If \(p > 3\) then let \(D\) be the hamiltonian cycle \(E_{2,0}\) of \(rt(2, p-1)\), and if \(p = 3\) then let \(D\) be the cycle induced by the nodes on row 2. If there is a cycle \(C\) then join \(C\) and \(D\) over two corresponding links and denote the new cycle by \(D\) also. Now join \(\rho(s, e)\) to the cycle \(D\), and the path obtained is as required.

**Sub-case (iii) s lies on row 0 and e lies on rows 2, 3, ....., p - 1.**

Suppose that \(p > 3\). If \(\{s, e\}\) is even then let the node \(e'\) on row 1 be such that \(e'\) and \(s\) have a common neighbour on row 0 and also such that \(e'\) does not lie on the same column as \(e\). If \(\{s, e\}\) is odd then let \(e'\) be the neighbour of \(s\) on row 1. By the construction in Sub-case (ii), there is a path \(\rho_A(s, e')\) in \(rt(0, 1)\) of length \(2k - \epsilon\).

Let \(s'\) be the neighbour of \(e'\) on row 2 (note that \(s' \neq e\) and that \(\{s', e\}\) is odd). By Lemma 3.3.2, there is a path \(\rho_X(s', e)\) in \(rt(2, p-1)\) of length \((p-2)k - 1\). The path

\[
\rho_A(s, e'), (e', s'), \rho_X(s', e)
\]
3.3. The base case

is as required.

Suppose that $p = 3$. Let $s'$ be a neighbour of $e$ on row 2 so that $s'$ does not lie on the same column as $s$, and let $e'$ be the neighbour of $s'$ on row 2. By the construction in Sub-case $(ii)$, there is a path $\rho_A(s, e')$ in $rt(0, 1)$ of length $2k - \epsilon$. Let $\rho_X(s', e)$ be the path on row 2 of length $k - 1$. The path

$$\rho_A(s, e'), (e', s'), \rho_X(s', e)$$

is as required.

Sub-case $(iv)$ $s$ and $e$ lie on row 1.

Let $s = v_{i,1}$ and $e = v_{j,1};$ w.l.o.g. we may assume that $i < j$. Let $\rho_1(s, e)$ be the path on row 1 containing the link $(v_{i,0}, v_{i,1})$. Join $\rho_1(s, e)$ to the cycle induced by the nodes on row 0 over the links $(v_{i,0}, v_{i,1})$ and $(v_{0,0}, v_{0,1})$, and denote the resulting path by $\rho_A(s, e)$. Iteratively join $\rho_A(s, e)$ to appropriate links $(v_{i,j}, v_{i,j+1})$ over $(v_{0,0}, v_{0,1})$ so that the nodes used on row 1 do not already appear on $\rho_A(s, e)$. Links should be replaced (by paths) so that if \{s, e\} is odd (resp. even) then every node of row 1 appears on the amended path (resp. except one). Denote the amended path by $\rho(s, e)$.

If $p \geq 4$ then let $D$ be the hamiltonian cycle $E_{2,1}$ of $rt(2, p - 1)$, and if $p = 3$ then let $D$ be the cycle induced by the nodes on row 2. Join $\rho(s, e)$ to $D$ over two corresponding links, and the resulting path is as required.

Sub-case $(v)$ $s$ lies on row 1 and $e$ lies on rows 2, 3, ..., $p - 1$.

Suppose that $p \geq 4$. Let $e'$ be a neighbour of $s$ on row 1 such that $e$ does not lie on the same column as $e'$. We now define a path $\rho_A(s, e')$ in $rt(0, 1)$. If $s = v_{1,1}$ and $e' = v_{1,0}$ then define $\rho_A(s, e')$ as

$$N^*(s, v_{1,k-1}), (v_{1,k-1}, v_{0,k-1}), (v_{0,k-1}, v_{0,0}), (v_{0,0}, e');$$

if $s = v_{1,0}$ and $e' = v_{1,1}$ then define $\rho_A(s, e')$ as

$$N^-(s, v_{1,2}), (v_{1,2}, v_{0,2}), (v_{0,2}, v_{0,1}), (v_{0,1}, e');$$

otherwise, let $\rho_1(s, e')$ be the path on row 1 containing the link $(v_{1,0}, v_{1,1})$, and join $\rho_1(s, e')$ to the cycle induced by the nodes on row 0 (which contains the faulty
3.3. The base case

link) over the links \((v_{1,0}, v_{1,1})\) and \((v_{0,0}, v_{0,1})\), denoting the resulting path by \(\rho_A(s, e')\) (joining as we do results in the path \(\rho_A(s, e')\) being fault-free).

Let \(s'\) be the neighbour of \(e'\) on row 2. By Lemma 3.3.2, there is a path \(\rho_X(s', e)\) in \(rt(2, p - 1)\) of length \((p - 2)k - \epsilon\). The path

\[\rho_A(s, e'), (e', s'), \rho_X(s', e)\]

is as required.

Suppose that \(p = 3\). Let \(e'\) be a node on row 1 such that \(s \neq e'\) and \(e'\) is in a column adjacent to the column on which \(e\) lies. Clearly, \(\{s, e\}\) is odd if, and only if, \(\{s, e'\}\) is odd (node \(e'\) and \(e\) have the same parity). We now build a path \(\rho_A(s, e')\) in \(rt(0, 1)\); w.l.o.g. we may assume that \(s = v_{1,i}, e' = v_{i,j}\) and \(i < j\), with \(i \neq 0\) (as usual, we can apply automorphisms of \(rt(0, 1)\) if necessary). If \(\{s, e\}\) is odd and \(i \neq 1\) then define \(\rho_A(s, e')\) as

\[C_1^-(s, v_{0,i}), Z^+(v_{0,i}, v_{0,j-1}), (v_{0,j-1}, v_{0,j}), C_0^+(v_{0,j}, e').\]

If \(\{s, e\}\) is odd and \(i = 1\) then define \(\rho_A(s, e')\) as

\[N^+(s, v_{1,j-1}), (v_{1,j-1}, v_{0,j-1}), (v_{0,j-1}, v_{0,j}), C_0^+(v_{0,j}, e').\]

If \(\{s, e\}\) is even and \(s \neq 1\) then define \(\rho_A(s, e')\) as

\[C_1^-(s, v_{0,i}), Z^+(v_{0,i}, v_{0,j-2}), (v_{0,j-2}, v_{0,j-1}), (v_{0,j-1}, v_{0,j}), C_0^+(v_{0,j}, e').\]

If \(\{s, e\}\) is even and \(s = 1\) then define \(\rho_A(s, e')\) as

\[N^+(s, v_{1,j-2}), (v_{1,j-2}, v_{0,j-2}), (v_{0,j-2}, v_{0,j-1}), (v_{0,j-1}, v_{0,j}), C_0^+(v_{0,j}, e').\]

Let \(s'\) be the neighbour of \(e'\) on row 2 and let \(\rho_X(s', e)\) be the path on row 2 of length \(k - 1\). The path

\[\rho_A(s, e'), (e', s'), \rho_X(s', e)\]

is as required.

Sub-case (vi) \(s\) and \(e\) lie on rows \(2, 3, \ldots, p - 1\).

Suppose that \(p \geq 4\). By Lemma 3.3.2, there is a path \(\rho_X(s, e)\) in \(rt(2, p - 1)\) of length \((p - 2)k - \epsilon\). Let \(C\) be the cycle

\[C_1^-(v_{1,0}, v_{0,0}), (v_{0,0}, v_{1,0}).\]
3.3. The base case

Joining \( \rho_X(s, e) \) to \( C \) over two corresponding links yields a required path.

Suppose that \( p = 3 \). If \( (s = v_{2,0} \text{ and } e = v_{2,1}) \) or \( (e = v_{2,0} \text{ and } s = v_{2,1}) \) then let \( \rho_X(s, e) \) be the path on row 2 of length \( k - 1 \); otherwise, let \( \rho_X(s, e) \) be the path on row 2 not containing the link \( (v_{2,0}, v_{2,1}) \). Join \( \rho_X(s, e) \) to \( C \) over two corresponding links and denote the resulting path by \( \rho(s, e) \).

If \( (s = v_{2,0} \text{ and } e = v_{2,1}) \) or \( (e = v_{2,0} \text{ and } s = v_{2,1}) \) then \( \rho(s, e) \) is as required. Otherwise, iteratively join \( \rho(s, e) \) to appropriate links \( (v_{2,1}, v_{2,l+1}) \) over \( (v_{1,l}, v_{1,l+1}) \) so that the nodes used on row 2 do not already appear on \( \rho(s, e) \). Links should be replaced (by paths) so that if \{s, e\} is odd (resp. even) then every node of row 2 appears on the amended path (resp. except one). The path so obtained is as required.

**Case (d)** The faulty link is \( (v_{a,0}, v_{a+1,0}) \), where \( 1 \leq a \leq p - 3 \).

**Sub-case (i)** \( s \) and \( e \) lie on rows \( 0, 1, \ldots, a + 1 \).

By Case (c), there is a path \( \rho_A(s, e) \) in \( rt(0, a+1) \) of length \( (a+2)k - \epsilon \). If \( a \neq p - 3 \) then let \( C \) be the hamiltonian cycle \( E_{a+2,0} \) of \( rt(a+2, p-1) \), and if \( a = p - 3 \) then let \( C \) be the cycle induced by the nodes on row \( p - 1 \). Joining \( \rho_A(s, e) \) and \( C \) over two corresponding links yields a path as required.

**Sub-case (ii)** \( s \) lies on rows \( 0, 1, \ldots, a + 1 \) and \( e \) lies on rows \( a + 2, a + 3, \ldots, p - 1 \).

Suppose that \( a \neq p - 3 \). Let the node \( e' \) on row \( a + 1 \) be such that \( s \neq e' \) and \( \{s, e\} = \{s, e'\} \). By Case (c), there is a path \( \rho(s, e') \) in \( rt(0, a + 1) \) of length \( (a+2)k - \epsilon \). Let \( s' \) be the node on row \( a + 2 \) adjacent to \( e' \). By Lemma 3.3.2, there is a path \( \rho_X(s', e) \) in \( rt(a + 2, p - 1) \) of length \( (p-a-2)k - 1 \). The path

\[
\rho_A(s, e'), (e', s'), \rho_X(s', e)
\]

is as required.

Suppose that \( a = p - 3 \). Let the node \( e' \) on row \( a + 1 \) be such that \( e' \neq s \) and \( e' \) lies on a column adjacent to the column on which \( e \) lies. By Case (c), there is a path \( \rho(s, e') \) in \( rt(0, p-2) \) of length \( (p-1)k - \epsilon \). Let \( s' \) be the neighbour of \( e' \) on row \( p - 1 \) and let \( \rho_X(s', e) \) be the path of length \( k - 1 \) on row \( p - 1 \). The path

\[
\rho_A(s, e'), (e', s'), \rho_X(s', e)
\]
3.3. The base case

is as required.

Proposition 3.3.4 Consider the k-ary 2-cube $Q_k^2$ where $k \geq 6$ is even and where 2 of the links are faulty. Let $s$ and $e$ be any two distinct nodes. There is a path of length $k^2 - 1$ (resp. $k^2 - 2$) from $s$ to $e$ if $\{s, e\}$ is odd (resp. even).

Proof: W.l.o.g. we may assume that $(v_{0,0}, v_{1,0})$ is a faulty link. Partition $Q_k^2$ into $rt(k - 1, 0)$ and $rt(1, k - 2)$. As usual, $\epsilon = 1$ if $\{s, e\}$ is odd, and $\epsilon = 2$ if $\{s, e\}$ is even.

Case (a) Both $s$ and $e$ lie in $rt(k - 1, 0)$.

By Proposition 3.3.3, there is a path $\rho_A(s, e)$ in $rt(k - 1, 0)$ of length $2k - \epsilon$. Either there is a link of $\rho_A(s, e)$ on row $k - 1$ that is not incident with any faulty link or there is a link of $\rho_A(s, e)$ on row 0 that is not incident with any faulty link; w.l.o.g. suppose that $(v_{k-1,i}, v_{k-1,i+1})$ is a link of $\rho_A(s, e)$ such that neither $(v_{k-1,i}, v_{k-2,i})$ nor $(v_{k-1,i+1}, v_{k-2,i+1})$ is faulty (the alternative case is similar). By Proposition 3.3.3, there is a path $\rho_X(v_{k-2,i}, v_{k-2,i+1})$ in $rt(1, k - 2)$ of length $(k - 2)k - 1$. The path obtained by joining $\rho_A(s, e)$ to $\rho_X(v_{k-2,i}, v_{k-2,i+1})$ over $(v_{k-1,i}, v_{k-1,i+1})$ is as required.

Case (b) $s$ lies in $rt(k - 1, 0)$ and $e$ lies in $rt(1, k - 2)$.

Let $(v_{k-1,i}, v_{k-2,i})$ be a healthy link such that $s \neq v_{k-1,i}, e \neq v_{k-2,i}$ and $\{s, v_{k-1,i}\} = \{s, e\}$. By Proposition 3.3.3, there is a path $\rho_A(s, v_{k-1,i})$ in $rt(k - 1, 0)$ of length $2k - \epsilon$ and there is a path $\rho_X(v_{k-2,i}, e)$ in $rt(1, k - 2)$ of length $(k - 2)k - 1$. The path

$\rho_A(s, v_{k-1,i}), (v_{k-1,i}, v_{k-2,i}), \rho_X(v_{k-2,i}, e)$

is as required.

Finally, we deal with the case when there is one faulty node and one faulty link.

Proposition 3.3.5 Consider the k-ary 2-cube $Q_k^2$ where $k \geq 6$ is even and where there is a faulty node and a faulty link. Let $s$ and $e$ be any two distinct, non-faulty nodes. There is a path of length at least $k^2 - 3$ (resp. $k^2 - 4$) from $s$ to $e$ if $\{s, e\}$ is odd (resp. even).

Proof: W.l.o.g. we may assume that the faulty node is $v_{0,0}$. Moreover, we may assume that either the faulty link does not lie in $rt(0, 1)$ or the faulty link is
(v_{0,0}, v_{0,1}) (again, by applying the usual automorphisms). However, if the faulty link is (v_{0,0}, v_{0,1}) then we can assume that there are no faulty links as the fact that v_{0,0} is a faulty node means that the link (v_{0,0}, v_{0,1}) is never used. Thus, we can assume that the faulty link does not lie in \( rt(0, 1) \). As usual, \( \epsilon = 1 \) if \( \{s, e\} \) is odd, and \( \epsilon = 2 \) if \( \{s, e\} \) is even.

**Case (a)** Both \( s \) and \( e \) lie in \( rt(0, 1) \).

By Lemma 3.3.1, there is a path \( \rho_A(s, e) \) in \( rt(0, 1) \) of length at least \( 2k - 2 - \epsilon \).

Either there is a link of \( \rho_A(s, e) \) on row 0 that is not incident with \( v_{0,0} \) nor a faulty link, or there is a link of \( \rho_A(s, e) \) on row 1 that is not incident with a faulty link. W.l.o.g. suppose that \( v_{1,i}, v_{1,i+1} \) is a link of \( \rho_A(s, e) \) that is not incident with a faulty link (the alternative case is similar). By Proposition 3.3.3, there is a path \( \rho_X(v_{2,i}, v_{2,i+1}) \) in \( rt(2, k - 1) \) of length \( (k - 2)k - 1 \). The path obtained by joining \( \rho_A(s, e) \) to \( \rho_X(v_{2,i}, v_{2,i+1}) \) over \( (v_{1,i}, v_{1,i+1}) \) is as required.

**Case (b)** \( s \) lies in \( rt(0, 1) \) and \( e \) lies in \( rt(2, k - 1) \).

Let \( v_{1,i} \) be such that \( s \neq v_{1,i}, (v_{1,i}, v_{2,i}) \) is healthy and \( \{s, v_{1,i}\} = \{s, e\} \). By Lemma 3.3.1, there is a path \( \rho_A(s, v_{1,i}) \) in \( rt(0, 1) \) of length at least \( 2k - 2 - \epsilon \).

By Proposition 3.3.3, there is a path \( \rho_X(v_{2,i}, e) \) in \( rt(2, k - 1) \) of length \( (k - 2)k - 1 \). The path
\[
\rho_X(s, v_{1,i}), (v_{1,i}, v_{2,i}), \rho_X(v_{2,i}, e)
\]
is as required. \( \square \)

From Propositions 3.3.1, 3.3.2, 3.3.4 and 3.3.5, we obtain the base case for our main result so long as \( k \geq 6 \). However, when \( k = 4 \) a simple computer program (implementing an exhaustive search) verifies that Propositions 3.3.1, 3.3.2, 3.3.4 and 3.3.5 all still hold (we leave this verification as an exercise). Hence, we have the following result.

**Theorem 3.3.3** Let \( k \geq 4 \) be even. In a faulty \( k \)-ary 2-cube \( Q_2^k \) in which the number of node faults \( f_v \) and the number of link faults \( f_e \) are such that \( f_v + f_e \leq 2 \), given any two healthy nodes \( s \) and \( e \) of \( Q_2^k \), there is a path from \( s \) to \( e \) of length at least \( k^2 - 2f_v - 1 \) (resp. \( k^2 - 2f_v - 2 \)) if the nodes \( s \) and \( e \) have different (resp. the same) parities.
3.4 The inductive step

In this section, we complete the proof by induction of our main theorem. The following lemma simplifies the situation considerably.

**Lemma 3.4.1** Let $Q_n^k$ have $2n - 2$ faulty nodes and links, where $n \geq 4$. There exists a dimension $d$ such that when we partition $Q_n^k$ over dimension $d$, the resulting $k$-ary $(n - 1)$-cubes $Q_{d,0}, Q_{d,1}, \ldots, Q_{d,k-1}$ each contain at most $2n - 4$ faulty nodes and links.

**Proof:** Suppose as our induction hypothesis that $n \geq 5$ and that the result holds for $Q_{n-1}^k$ (with $2n - 4$ faults). Let $Q_n^k$ have $2n - 2$ faults. Partition $Q_n^k$ over dimension 1; if the resulting $k$-ary $(n - 1)$-cubes $Q_{1,0}, Q_{1,1}, \ldots, Q_{1,k-1}$ are such that each contains at most $2n - 4$ faults then we are done. So w.l.o.g. suppose that $Q_{1,0}$ contains $2n - 2$ or $2n - 3$ faults.

Suppose that $Q_{1,0}$ contains $2n - 3$ faults, and so there is exactly 1 fault not in $Q_{1,0}$. Temporarily regard some fault, $w$, say, of $Q_{1,0}$ as healthy and apply the induction hypothesis to $Q_{1,0}$ (note that $w$ might be a node or a link). Thus, there is a dimension $d$ such that when we partition $Q_{1,0}$ over dimension $d$, the resulting $k$-ary $(n - 2)$-cubes each contain at most $2n - 6$ faults. Consequently, when we partition $Q_n^k$ over dimension $d$, each of the resulting $k$-ary $(n - 1)$-cubes contains at most $2n - 4$ faults (the ‘temporarily healthy fault’ $w$ needs to be recast as faulty, and there is 1 other fault not in $Q_{1,0}$ to consider).

Suppose that $Q_{1,0}$ contains $2n - 2$ faults, and so there are no faults outside $Q_{1,0}$. Temporarily regard 2 faults, $w$ and $w'$, say, of $Q_{1,0}$ as healthy and apply the induction hypothesis to $Q_{1,0}$. Thus, there is a dimension $d$ such that when we partition $Q_{1,0}$ over dimension $d$, the resulting $k$-ary $(n - 2)$-cubes each contain at most $2n - 6$ faults. Consequently, when we partition $Q_n^k$ over dimension $d$, each of the resulting $k$-ary $(n - 1)$-cubes contains at most $2n - 4$ faults (the 2 ‘temporarily healthy faults’ $w$ and $w'$ need to be recast as faulty).

In order for the result to follow by induction, all we need to do is to verify the statement of the lemma for when $n = 4$. Let the faults of $Q_4^k$ be $w_i$, for $i = 1, 2, \ldots, 6$. Partition $Q_4^k$ over dimension 1. Either each resulting $k$-ary 3-cube contains at most
4 faults, and we are done, or the nodes involved in at least 5 of \( \{w_i : i = 1, 2, \ldots, 6\} \) have identical fourth components (if \( w_i \) is a link then the nodes involved in \( w_i \) are the nodes of the link, and if \( w_i \) is a node then the node involved in \( w_i \) is the node itself). We may assume that it is the latter and that the 5 faults whose fourth components (of the nodes involved) are identical are \( w_1, w_2, w_3, w_4 \) and \( w_5 \).

Partition \( Q^4 \) over dimension 2. Either each resulting \( k \)-ary 3-cube contains at most 4 faults, and we are done, or one of the resulting \( k \)-ary 3-cubes contains either 5 or 6 faults. We may assume that the third components of \( w_1, w_2, w_3 \) and \( w_4 \) are identical.

Partition \( Q^5 \) over dimension 3. Either each resulting \( k \)-ary 3-cube contains at most 4 faults, and we are done, or one of the resulting \( k \)-ary 3-cubes contains either 5 or 6 faults. We may assume that the second components of \( w_1, w_2 \) and \( w_3 \) are identical.

Partition \( Q^6 \) over dimension 4. Either each resulting \( k \)-ary 3-cube contains at most 4 faults, and we are done, or one of the resulting \( k \)-ary 3-cubes contains either 5 or 6 faults. We may assume that the first components of \( w_1 \) and \( w_2 \) are identical. This yields a contradiction as either: \( w_1 \) and \( w_2 \) are nodes and \( w_1 \neq w_2 \); or \( w_1 \) or \( w_2 \) is a link joining a node to itself. The result follows.

Let us reexamine the proof of Lemma 3.4.1. Ideally we would like Lemma 3.4.1 to apply when \( n = 3 \) but the argument in the proof fails. However, we can classify exactly the fault configurations leading to failure.

Suppose that \( Q^3 \) has 4 faulty nodes. Following through the argument in the proof of Lemma 3.4.1 yields that, up to isomorphism, the situations where the argument fails is when the 4 faults are of the form \((0,0,0), (a,0,0), (0,b,0)\) and \((0,0,c)\), for some \( a, b \) and \( c \) all different from 0.

Suppose that \( Q^3 \) has 3 faulty nodes and 1 faulty link. W.l.o.g. suppose that the faulty link lies in dimension 3. Following the argument in Lemma 3.4.1 yields that, up to isomorphism, the situations where the argument fails is when the 3 faulty nodes are of the form \((0,0,0), (0,b,0)\) and \((0,0,c)\), for some \( b \) and \( c \) different from 0, and the faulty link is of the form \(((a,0,0), (a+1,0,0))\), for some \( a \).
3.4. The inductive step

Suppose that $Q^k_3$ has 2 faulty nodes and 2 faulty links. W.l.o.g. suppose that one of the faulty links lies in dimension 3 with the other in dimension 2 (the two links cannot lie in the same dimension as otherwise we could partition over this dimension and be done). Following the argument in Lemma 3.4.1 yields that, up to isomorphism, the situations where the argument fails is when the 2 faulty nodes are of the form $(0,0,0)$ and $(0,0,c)$, for some $c$ different from 0, and the faulty links are of the form $((a,0,0),(a+1,0,0))$ and $((0,b,0),(0,b+1,0))$, for some $a$ and $b$.

Suppose that $Q^k_3$ has 1 faulty node and 3 faulty links. W.l.o.g. suppose that one of the faulty links lies in dimension 1, one in dimension 2 and one in dimension 3. Following the argument in Lemma 3.4.1 yields that, up to isomorphism, the situations where the argument fails is when the faulty node is of the form $(0,0,0)$ and the faulty links are of the form $((a,0,0),(a+1,0,0)), ((0,b,0),(0,b+1,0))$ and $((0,0,c),(0,0,c+1))$, for some $a$, $b$ and $c$.

Suppose that $Q^k_3$ has 4 faulty links. In this case, Lemma 3.4.1 holds as at least 2 faulty links lie in the same dimension and we can partition over this dimension. We shall use these observations in the proof of the following theorem.

Throughout the rest of the chapter, we adopt the following notation. Suppose that we partition $Q_3^n$ over some dimension $d$ to get the $k$-ary $(n-1)$-cubes $Q_d, Q_{d,1}, \ldots, Q_{d,k-1}$. Let $x$ be a node of $Q_{d,i}$, say. Then we refer to the node in $Q_{d,j}$ corresponding to $x$ (that is, the node of $Q_{d,j}$ whose name is identical to that of $x$ except that its component on dimension $d$ is $j$ as opposed to $i$) as $x_j$. We also refer to the node $x$ as $x_i$.

**Theorem 3.4.2** Let $Q_n^k$ be a $k$-ary $n$-cube, for some $n \geq 2$ and some even $k \geq 4$, with $f_v$ faulty nodes and $f_e$ faulty links, where $0 \leq f_v + f_e \leq 2n - 2$. If $s$ and $e$ are distinct healthy nodes and $\{s, e\}$ is odd (resp. even) then there exists a path from $s$ to $e$ of length at least $k^n - 2f_v - 1$ (resp. $k^n - 2f_v - 2$).

**Proof:** We proceed by induction on $n$. The base case of the induction is handled by Theorem 3.3.3. Suppose, as our induction hypothesis, that the result holds for $Q_m^k$, where $n \geq 3$ and for all $m < n$. Let $Q_n^k$ be a $k$-ary $n$-cube as in the statement of the theorem. Throughout, $\epsilon = 1$ if $\{s, e\}$ is odd, and $\epsilon = 2$ if $\{s, e\}$ is even.
Suppose that \( n \geq 4 \). By Lemma 3.4.1, we may assume that when we partition \( Q_n^k \) over dimension 1, the resulting \( k \)-ary \((n - 1)\)-cubes \( Q_{1,0}, Q_{1,1}, \ldots, Q_{1,k-1} \) each contain at most \( 2n - 4 \) faults. Suppose that the number of faulty nodes in \( Q_{1,i} \) is \( f_i \), for \( i = 0, 1, \ldots, k - 1 \).

**Case (a) s and e lie in \( Q_{1,0} \).**

By the induction hypothesis, there is a path \( \rho_0(s, e) \) in \( Q_{1,0} \) of length at least \( k^{n-1} - 2f_0 - 1 \). Let \( \langle w_0, z_0 \rangle \) be a link of \( \rho_0(s, e) \) for which \( w_1 \) and \( z_1 \) are healthy nodes (of \( Q_{1,1} \)) and \( \langle w_0, w_1 \rangle \) and \( \langle z_0, z_1 \rangle \) are healthy links (a simple counting argument shows the existence of such a link; if otherwise, there is no such edge, i.e., for \( \forall \langle w_0, z_0 \rangle \in \rho_0(s, e) \), either \( \langle w_1, z_1 \rangle \) or \( \langle w_0, w_1 \rangle \) or \( \langle z_0, z_1 \rangle \) is faulty; then the number of faulty edges must be at least half of the path length; as there are at most \( 2n - 2 \) faults in \( Q_{n,k} \) and \( k \geq 4, n \geq 3 \), we have a contradiction). By the induction hypothesis, there is a path \( \rho_1(w_1, z_1) \) in \( Q_{1,1} \) of length at least \( k^{n-1} - 2f_1 - 1 \). Let \( \rho(s, e) \) be the join of \( \rho_0(s, e) \) to \( \rho_1(w_1, z_1) \) over \( \langle w_0, z_0 \rangle \). The path \( \rho(s, e) \) has length at least \( 2k^{n-1} - 2(f_0 + f_1) - 1 \). Proceeding similarly and iteratively with appropriate paths in \( Q_{1,2}, Q_{1,3}, \ldots, Q_{1,k-1} \) yields a path from \( s \) to \( e \) of the required length.

**Case (b) s lies in \( Q_{1,0} \) and e lies in \( Q_{1,a} \), for \( a \neq 0 \).**

A simple counting argument yields that there exists a healthy node \( w_0 \in Q_{1,0} \setminus \{ e_0 \} \) such that: \( \{ s, w_0 \} \) is odd; \( w_i \) is healthy, for all \( i = 0, 1, \ldots, k - 1 \); and all links of \( \{(w_i, w_{i+1}) : i = 0, 1, \ldots, k - 2\} \cup \{(w_{k-1}, w_0)\} \) are healthy. By the induction hypothesis, there exists a path \( \rho_0(s, w_0) \) in \( Q_{1,0} \) of length at least \( k^{n-1} - 2f_0 - 1 \).

Suppose that \( a \neq 1 \). A simple counting argument yields that there exists a healthy node \( z_1 \in Q_{1,1} \setminus \{ e_1 \} \) such that: \( \{ w_1, z_1 \} \) is odd; \( z_i \) is healthy, for all \( i = 0, 1, \ldots, k - 1 \); and all links of \( \{(z_i, z_{i+1}) : i = 0, 1, \ldots, k - 2\} \cup \{(z_{k-1}, z_0)\} \) are healthy. By the induction hypothesis, there exists a path \( \rho_1(w_1, z_1) \) in \( Q_{1,1} \) of length at least \( k^{n-1} - 2f_1 - 1 \). Denote the path

\[ \rho_0(s, w_0), (w_0, w_1), \rho_1(w_1, z_1) \]

by \( \rho(s, z_1) \).
Suppose that \( a \neq 2 \). By the induction hypothesis, there exists a path \( \rho_2(z_2, w_2) \) in \( Q_{1,2} \) of length at least \( k^{n-1} - 2f_2 - 1 \). Denote the path

\[
\rho(s, z_1), (z_1, z_2), \rho_2(z_2, w_2)
\]

by \( \rho(s, w_2) \).

Proceeding iteratively in this way yields a path \( \rho(s, z_{a-1}) \) or \( \rho(s, w_{a-1}) \), depending upon whether \( a - 1 \) is odd or even, respectively, of length at least \( ak^{n-1} - 2(f_0 + f_1 + \ldots + f_{a-1}) - 1 \). W.l.o.g., suppose that the path is \( \rho(s, z_{a-1}) \) (the other case is similar). The node \( z_a \) is odd if, and only if, the node \( s \) is odd; hence, \( \{s, e\} = \{z_a, e\} \).

By the induction hypothesis, there exists a path \( \rho_a(z_a, e) \) in \( Q_{1,a} \) of length at least \( k^{n-1} - 2f_a - \epsilon \). Denote the path

\[
\rho(s, z_{a-1}), (z_{a-1}, z_a), \rho_a(z_a, e)
\]

by \( \rho'(s, e) \). The path \( \rho'(s, e) \) has length at least \( (a + 1)k^{n-1} - 2(f_0 + f_1 + \ldots + f_a) - \epsilon \).

A simple counting argument yields that there is a link \( (x_a, y_a) \) of \( \rho_a(z_a, e) \) such that \( x_{a+1} \) and \( y_{a+1} \) are both healthy nodes and \( (x_a, x_{a+1}) \) and \( (y_a, y_{a+1}) \) are both healthy links (to see this, note that \( \rho_a(z_a, e) \) has length at least \( k^{n-1} - 2f_a - \epsilon \geq 2^{2n-2} - 2(2n - 4) - 2 = 2^{2n-2} - 4n + 6 \), and so there are at least \( 2^{2n-3} - 2n + 3 \) mutually disjoint links on \( \rho_a(z_a, e) \); as there are at most \( 2n - 2 \) faulty links in our \( Q^k_n \) and \( 2^{2n-3} - 2n + 3 > 2n - 2 \), when \( n \geq 3 \), at least one such link \( (x_a, y_a) \) of \( \rho_a(z_a, e) \) must be as required). By the induction hypothesis, there is a path \( \rho_{a+1}(x_{a+1}, y_{a+1}) \) in \( Q_{1,a+1} \) of length at least \( k^n - 2f_{a+1} - 1 \). Form the path obtained by joining \( \rho'(s, e) \) to \( \rho_{a+1}(x_{a+1}, y_{a+1}) \) over \( (x_a, y_a) \) and denote this path by \( \rho''(s, e) \). The path \( \rho''(s, e) \) has length at least \( (a + 2)k^{n-1} - 2(f_0 + f_1 + \ldots + f_{a+1}) - \epsilon \). Proceeding similarly and iteratively in \( Q_{1,a+2}, Q_{1,a+3}, \ldots, Q_{1,k-1} \) results in a path from \( s \) to \( e \) of the required length (the construction can be visualized as in Fig.3.7).

Now suppose that \( n = 3 \) and suppose further that we have no faulty links (we deal with when there are faulty links later). From the observation following Lemma 3.4.1, we may assume that we have 4 faulty nodes and that these nodes are \( (0,0,0) \), \( (a,0,0) \), \( (0,b,0) \) and \( (0,0,c) \), for some \( a, b \) and \( c \) all different from 0; otherwise the construction above in Cases \( (a) \) and \( (b) \) can be used to build our path.
3.4. The inductive step

Partition $Q^k_3$ over dimension 1 to obtain the $k$-ary 2-cubes $Q_{1,0}, Q_{1,1}, \ldots, Q_{1,k-1}$; note that $(0,0,0)$, $(a,0,0)$ and $(0,b,0)$ lie in $Q_{1,0}$.

Case (c) $s$ and $e$ lie in $Q_{1,0}$.

Temporarily suppose that $(0,0,0)$ is healthy. By Theorem 3.3.3, there is a path $\rho_0(s,e)$ in $Q_{1,0}$ of length at least $k^2 - 4 - \epsilon$ but upon which $(0,0,0)$ may lie. If $(0,0,0)$ lies on $\rho_0(s,e)$ then choose $y_0 = (0,0,0)$, otherwise choose $y_0$ to be any node of $\rho_0(s,e)$ different from $s$ and $e$.

Let $y_0^-$ and $y_0^+$ be the nodes immediately before and after $y_0$, respectively, on $\rho_0(s,e)$. W.l.o.g., we may suppose that $y_{k-1}^-$ and $y_1^+$ are healthy nodes (and that $(y_0^-, y_{k-1}^-)$ and $(y_0^+, y_1^+)$ are healthy links; recall, there is 1 faulty node outside $Q_{1,0}$).

A simple counting argument yields that there exists a healthy node $w_{k-1} \in Q_{1,k-1} \setminus \{y_{k-1}^\} \text{ such that } \{y_{k-1}^-, w_{k-1}\}$ is odd and $w_i$ is healthy, for all $i = 1, 2, \ldots, k-1$ (and the links of $\{(w_i, w_{i+1}) : i = 0, 1, \ldots, k-2\}$ are healthy; to see this, note that there are at least $\lceil (k^2 - 1)/2 \rceil$ healthy nodes $w_{k-1}$ for which $\{y_{k-1}^-, w_{k-1}\}$ is odd, and this number is greater than 0). By Theorem 3.3.3, there exists a path $\rho_{k-1}(y_{k-1}^-, w_{k-1})$ in $Q_{1,k-1}$ of length at least $k^2 - 2f_{k-1} - 1$.

A simple counting argument yields that there exists a healthy node $z_{k-2} \in Q_{1,k-2} \setminus \{y_{k-2}^+, w_{k-2}\}$ such that $\{w_{k-2}, z_{k-2}\}$ is odd and $z_i$ is healthy, for all $i = 1, 2, \ldots, k-1$ (and the links of $\{(z_i, z_{i+1}) : i = 0, 1, \ldots, k-3\}$ are healthy). By Theorem 3.3.3, there exists a path $\rho_{k-2}(w_{k-2}, z_{k-2})$ in $Q_{1,k-2}$ of length at least $k^2 - 2f_{k-2} - 1$.

![Figure 3.7: The construction in Case (b).](image)
3.4. The inductive step

Proceeding iteratively in this way yields a path $\rho'(s, z_1)$ defined as

$$
\rho(s, y_0^-), (y_0^-, y_{k-1}^-), \rho_{k-1}(y_{k-1}^-, w_{k-1}), (w_{k-1}, w_{k-2}), \\
\rho_{k-2}(w_{k-2}, z_{k-2}), (z_{k-2}, z_{k-3}), \ldots, (z_2, z_1).
$$

By Theorem 3.3.3, there is a path $\rho_1(z_1, y_1^+)$ in $Q_{1,1}$ of length at least $k^2 - 2f_1 - 2$.

Consider the path $\rho''(s, e)$ defined as

$$
\rho'(s, z_1), \rho_1(z_1, y_1^+), (y_1^+, y_0^+), \rho_0(y_0^+), e.
$$

The length of this path is $k^3 - 2\sum_{i=1}^{k-1} f_i - 6 - \epsilon = k^3 - 8 - \epsilon$. Hence, the path $\rho''(s, e)$ is as required (the construction can be visualized as in Fig.3.8).

Case (d) $s$ lies in $Q_{1,0}$ and $e$ does not lie in $Q_{1,0}$.

For the moment, regard the node $x_0 = (0, 0, 0)$ as healthy. By Theorem 3.3.3, there is a path $\rho_0(s, x_0)$ in $Q_{1,0}$ of length at least $k^2 - 5$, if $\{s, x_0\}$ is odd, and $k^2 - 6$, if $\{s, x_0\}$ is even. Let $w_0$ be the node of $\rho_0(s, x_0)$ adjacent to $x_0$. W.l.o.g. we may assume $w_1$ and $(w_0, w_1)$ are healthy. There are two possibilities: either $e \in Q_{1,1}$ or $e \in Q_{1,m}$, where $0 \neq m \neq 1$.

Suppose that $e \in Q_{1,1}$ and $w_1 = e$. A simple counting argument yields that there exists a link $(y_0, z_0)$ of $\rho_0(s, w_0)$ such that $y_0 \neq w_0 \neq z_0$ and $y_1, z_1, (y_0, y_1)$ and $(z_0, z_1)$ are healthy. By Theorem 3.3.3, there is a path $\rho_1(y_1, z_1)$ in $Q_{1,1}$ that avoids $e$ and is of length at least $k^2 - 2(f_1 + 1) - 1$. Let $\rho(s, e)$ be the path obtained by joining

$$
\rho_0(s, w_0), (w_0, e).
$$

Figure 3.8: The construction in Case (c).
3.4. The inductive step

to $\rho_1(y_1, z_1)$ over the link $(y_0, z_0)$. As $\{s, x_0\} = \{s, e\}$, the length of $\rho(s, e)$ is at least $2k^2 - 2f_1 - 6 - \varepsilon$.

Suppose that $e \in Q_{1,1}$ and $w_1 \neq e$. By Theorem 3.3.3, there is a path $\rho_1(w_1, e)$ in $Q_{1,1}$ of length at least $k^2 - 2f_1 - 1$, if $\{w_1, e\}$ is odd, and $k^{n-1} - 2f_1 - 2$, if $\{w_1, e\}$ is even. Define the path $\rho(s, e)$ as

$$\rho_0(s, w_0), (w_0, w_1), \rho_1(w_1, e).$$

If $\{s, e\}$ is odd then $\{s, x_0\} = \{s, w_1\} \neq \{w_0, e\}$ and the length of $\rho(s, e)$ is at least $2k^2 - 2f_1 - 7$. If $\{s, e\}$ is even then $\{s, x_0\} = \{s, w_1\} = \{w_0, e\}$ and the length of $\rho(s, e)$ is at least $2k^2 - 2f_1 - 8$.

Hence, if $e \in Q_{1,1}$ then we have a path $\rho(s, e)$ in $Q_{1,0} \cup Q_{1,1}$ of length at least $2k^2 - 2f_1 - 6 - \varepsilon$ (the constructions can be visualized as in Fig.3.9).

A simple counting argument yields that there is a link $(u_1, v_1)$ of $\rho(s, e)$ such that $(u_1, u_2)$ and $(v_1, v_2)$ are both healthy. By Theorem 3.3.3, there is a path $\rho_2(u_2, v_2)$ in $Q_{1,2}$ of length at least $k^2 - 2f_2 - 1$. Join the path $\rho(s, e)$ to the path $\rho_2(u_2, v_2)$ over the link $(u_1, v_1)$ and denote the resulting path by $\rho(s, e)$ also. Proceeding iteratively in this way in $Q_{1,3}, Q_{1,4}, \ldots, Q_{1,k-1}$ yields a path $\rho(s, e)$ whose length is at least $k^3 - 2\sum_{i=1}^{k-1} f_i - 6 - \varepsilon = k^3 - 8 - \varepsilon$. Hence, the path $\rho(s, e)$ is as required.

Alternatively, suppose that $e \in Q_{1,m}$ where $0 \neq m \neq 1$. Let $y_1 \in Q_{1,1}$ be such that: $\{s, y_1\}$ is odd; $y_m \neq e$; and $y_i$ is healthy, for $i = 1, 2, \ldots, k - 1$ (and the links of $\{(y_i, y_{i+1}) : i = 1, 2, \ldots, k - 2\}$ are healthy). By the construction above, there is a path $\rho'(s, y_1)$ in $Q_{1,0} \cup Q_{1,1}$ of length $2k^2 - 2f_1 - 7$. 

Figure 3.9: The constructions in Case (d) when $e \in Q_{1,1}$. 


3.4. The inductive step

Suppose that \( m \neq 2 \). Let \( z_2 \in Q_{1,2} \) be such that: \( \{z_2, y_2\} \) is odd; \( z_a \neq e \); and \( z_i \) is healthy, for \( i = 1, 2, \ldots, k-1 \) (and the links of \( \{(z_i, z_{i+1}) : i = 2, 3, \ldots, k-2\} \) are healthy). By Theorem 3.3.3, there is a path \( \rho_2(y_2, z_2) \) in \( Q_{1,2} \) of length \( k^2 - 2f_2 - 1 \).

Suppose that \( m \neq 3 \). By Theorem 3.3.3, there is a path \( \rho_3(z_3, y_3) \) in \( Q_{1,3} \) of length \( k^2 - 2f_3 - 1 \). Proceeding in this way, we obtain paths \( \rho_2(y_2, z_2), \rho_3(z_3, y_3), \ldots, \) and so on until \( \rho_{m-1}(y_{m-1}, z_{m-1}) \), if \( m \) is odd, or \( \rho_{m-1}(z_{m-1}, y_{m-1}) \), if \( m \) is even. Applying Theorem 3.3.3 again yields a path \( \rho_m(z_m, e) \) or \( \rho_m(y_m, e) \) in \( Q_{1,m} \), depending upon whether \( m \) is odd or even, respectively. If \( m \) is odd (resp. even) then \( \rho_m(z_m, e) \) (resp. \( \rho_m(y_m, e) \)) has length at least \( k^2 - 2f_m - 1 \) if \( \{z_m, e\} \) (resp. \( \{y_m, e\} \)) is odd, and \( k^2 - 2f_m - 2 \) if \( \{z_m, e\} \) (resp. \( \{y_m, e\} \)) is even.

If \( m \) is odd then let \( \rho(s, e) \) be defined as

\[
\rho'(s, y_1), (y_1, y_2), \rho_2(y_2, z_2), (z_2, z_3), \rho_3(z_3, y_3), \ldots, (z_{m-1}, z_m), \rho_m(z_m, e),
\]

and if \( m \) is even then let \( \rho(s, e) \) be defined as

\[
\rho'(s, y_1), (y_1, y_2), \rho_2(y_2, z_2), (z_2, z_3), \rho_3(z_3, y_3), \ldots, (y_{m-1}, y_m), \rho_m(y_m, e).
\]

It can easily be verified that if \( m \) is odd then \( \{s, e\} = \{z_m, e\} \), and if \( m \) is even then \( \{s, e\} = \{y_m, e\} \). Thus, the length of the path \( \rho(s, e) \) is at least \((m + 1)k^2 - 2\sum_{i=0}^{m} f_i - 6 - \epsilon\). If \( m \neq k - 1 \) then the path \( \rho(s, e) \) can be iteratively joined to a path in \( Q_{1,i} \) of length \( k^2 - 2f_i - 1 \), for \( i = m + 1, m + 2, \ldots, k - 1 \), just as we did above, to obtain a path, also denoted \( \rho(s, e) \), of length at least \( k^3 - 2\sum_{i=0}^{k-2} f_i - 6 - \epsilon \).

Hence, our path \( \rho(s, e) \) is as required.

Case (e) \( s \) and \( e \) lie in \( Q_{1,p} \) and \( Q_{1,m} \), respectively, where \( m \neq 0 \neq p \neq m \).

W.l.o.g. suppose that \( p > m \). Let \( s' \in Q_{1,0} \) be such that \( s', s'_{k-1} \) and \( (s'_{k-1}, s') \) are healthy and \( \{s', s\} \) is odd. By the construction in Case (d), above, there is a path \( \rho'(s', e) \) in \( Q_{1,0} \cup Q_{1,1} \cup \ldots \cup Q_{1,m} \) of length at least \((m + 1)k^2 - 2\sum_{i=0}^{m} f_i - 7 \).

Let \( w_p \) be a node of \( Q_{1,p} \) such that: \( \{s, w_p\} \) is odd; \( w_0 \neq s' \); and \( w_i \) is healthy, for \( i = p, p + 1, \ldots, k - 1 \) (and the links of \( \{(w_i, w_{i+1}) : i = p, p + 1, \ldots, k - 2\} \) are healthy). By Theorem 3.3.3, there is a path \( \rho_p(s, w_p) \) in \( Q_{1,p} \) of length at least \( k^2 - 2f_p - 1 \).

Let \( y_{p+1} \) be a node of \( Q_{1,p+1} \) such that: \( \{w_{p+1}, y_{p+1}\} \) is odd; \( y_0 \neq s' \); and \( y_i \) is healthy, for \( i = p + 1, p + 2, \ldots, k - 1 \) (and the links of \( \{(y_i, y_{i+1}) : i = p + 1, p + 2, \ldots, k - 1\} \)
2, \ldots, k - 2} are healthy. By Theorem 3.3.3, there is a path \( \rho_{p+1}(w_{p+1}, y_{p+1}) \) in \( Q_{1,p+1} \) of length at least \( k^2 - 2f_{p+1} - 1 \).

Again, by Theorem 3.3.3, there are paths \( \rho_{p+2}(y_{p+2}, w_{p+2}), \rho_{p+3}(w_{p+3}, y_{p+3}) \), and so on, up to \( \rho_{k-2}(w_{k-2}, y_{k-2}) \), if \( p \) is even, and \( \rho_{k-2}(w_{k-2}, y_{k-2}) \), if \( p \) is odd, of lengths \( k^2 - 2f_{p+2} - 1, k^2 - 2f_{p+3} - 1, \ldots, k^2 - 2f_{k-2} - 1 \), respectively; note that \( \{s,e\} = \{w_{k-1}, s'_{k-1}\} \), if \( p \) is odd (resp. \( \{s,e\} = \{y_{k-1}, s'_{k-1}\} \), if \( p \) is even). Yet again, by Theorem 3.3.3, there is a path \( \rho_{k-1}(w_{k-1}, s'_{k-1}) \) (resp. \( \rho_{k-1}(y_{k-1}, s'_{k-1}) \)) in \( Q_{1,k-1} \) of length at least \( k^2 - 2f_{k-1} - \epsilon \), if \( p \) is even (resp. odd). Let \( \rho(s,e) \) be the path

\[
\rho_p(s, w_p), (w_p, w_{p+1}), \rho_{p+1}(w_{p+1}, y_{p+1}), (y_{p+1}, y_{p+2}), \ldots, (s'_{k-1}, s'), \rho'(s', e).
\]

The path \( \rho(s,e) \) has length at least \( (k - p + m - 1)k^2 - 2\sum_{i=0}^{m} f_i - 2\sum_{i=p}^{k-2} f_i - 2 - \epsilon \).

If \( p \neq m + 1 \) then the path \( \rho(s,e) \) can be iteratively joined to a path in \( Q_{1,i} \) of length \( k^2 - 2f_i - 1 \), for \( i = m + 1, m + 2, \ldots, p - 1 \), just as we did in Case (d), to obtain a path, also denoted \( \rho(s,e) \), of length at least \( k^3 - 2\sum_{i=1}^{k-1} f_i - 6 - \epsilon \). Hence, our path \( \rho(s,e) \) is as required.

**Case (f)** \( s \) and \( e \) lie in \( Q_{1,m} \) where \( m \neq 0 \).

By Theorem 3.3.3, there is a path \( \rho_m(s,e) \) in \( Q_{1,m} \) of length at least \( k^2 - 2f_m - \epsilon \).

There exists a link \((w_m, y_m)\) of \( \rho_m(s,e) \) such that \( u_{m+1}, y_{m+1}, (w_m, w_{m+1}) \) and \((y_m, y_{m+1})\) are healthy. By Theorem 3.3.3, there exists a path \( \rho_{m+1}(w_{m+1}, y_{m+1}) \) in \( Q_{1,m+1} \) of length at least \( k^2 - 2f_{m+1} - 1 \). Join \( \rho_m(s,e) \) to \( \rho_{m+1}(w_{m+1}, y_{m+1}) \) over \((w_m, y_m)\) and denote this path by \( \rho(s,e) \) also. The path \( \rho(s,e) \) can be iteratively joined to a path in \( Q_{1,i} \) of length \( k^2 - 2f_i - 1 \), for \( i = m + 2, m + 3, \ldots, m - 1 \) to obtain a path of length at least \( k^3 - 8 - \epsilon \) as required.

Now suppose that we have 1 faulty link. Partition over the dimension containing this faulty link and if each resulting \( k \)-ary 2-cube \( Q_{1,0}, Q_{1,1}, \ldots, Q_{1,k-1} \) contains at most 2 faults then apply the construction as in Cases (a) and (b) to build our path. Hence, we may assume that \( Q_{1,0} \) contains 3 faulty nodes. However, if we follow exactly the constructions in each of Case (c), (d), (e) and (f), then these constructions still apply and we obtain a path of the required length. Exactly the
3.5. Conclusions

same can be said of the scenarios when we have 2 and 3 faulty links. The result now follows.

We note that given $Q_n^k$, where $k \geq 4$ is even, and $f_v$ and $f_e$, where $f_v + f_e \leq 2n - 2$, there are configurations of $f_v$ faulty nodes, $f_e$ faulty links and pairs of distinct, healthy nodes so that the longest path joining the two nodes has length exactly $k^n - 2f_v - 1$ (resp. $k^n - 2f_v - 2$) if the parities of the two nodes are different (resp. the same). Hence, in this sense our result can be viewed as optimal.

Also, there are configurations of $2n - 1$ faulty nodes in $Q_n^k$ and pairs of healthy nodes such that the longest path joining the two nodes has length 1; take healthy, adjacent nodes $x$ and $y$ where all other neighbours of $x$ are faulty. Hence, the total number of faults in Theorem 3.4.2 cannot be increased.

3.5 Conclusions

Theorem 3.4.2, allied with the result in [171], fully resolves the situation as regards the existence of longest cycles in $k$-ary $n$-cubes where the total number of faults (nodes and links) is at most $2n - 2$ and where the faults are configured in a 'worst case' scenario with respect to the pair of nodes in question.

Of course, there are configurations of, for example, $2n - 2$ faulty nodes in $Q_n^k$ where certain pairs of nodes have paths joining them of lengths strictly greater than the bounds stated in Theorem 3.4.2. It would be interesting to build longest paths joining pairs of nodes but taking into account the configuration of faults (though this would appear to be a demanding task).

We expect that if we assume the conditional fault assumption then we should be able to tolerate more faults yet still prove a result analogous to Theorem 3.4.2. It would be worthwhile to investigate this scenario and we conjecture that the path lengths will be exactly as in Theorem 3.4.2.

The existence of paths and cycles in (faulty) interconnection networks does not guarantee that we can efficiently construct these paths and cycles using a distributed algorithm implemented on the underlying topology (see [149] as regards the issues involved with the distributed embedding of a hamiltonian cycle in a faulty $k$-ary
3.5. Conclusions

The existence of an efficient distributed algorithm which 'implements' Theorem 3.4.2 should be investigated.
Chapter 4

Bipanconnectivity and bicyclicity of \( k \)-ary \( n \)-cube

4.1 Introduction

Of interest to us in this chapter are the different paths and cycles embedded within \( k \)-ary \( n \)-cubes. Particularly, we are interested in questions relating to hamiltonicity, pancyclicity, panconnectivity, bipancyclicity and bipanconnectivity. These properties can be described as 'strong hamiltonicity' properties and their existence in an interconnection network enables a much higher degree of flexibility with regard to the simulation of linear arrays of processors or cycles of processors.

The notions in the preceding paragraph have been investigated in the context of a number of interconnection networks: for example, in crossed cubes \([54, 170]\), Möbius cubes \([78]\), augmented cubes \([118]\), alternating group graphs \([31]\), star graphs \([169]\), bubble-sort graphs \([97]\), and in hypercubes and hypercube-like networks \([55, 110, 131, 154, 156, 167, 168]\). As regards \( k \)-ary \( n \)-cubes, these notions have been considered in \([82, 163]\). In particular, it was proven in \([163]\): that \( Q_k^2 \) is almost-hamiltonian connected, bipanconnected and bipancyclic; that \( Q_n^k \) is almost-hamiltonian connected, for any \( k \); and that \( Q_n^k \) is hamiltonian-connected, for odd \( k \). Recently, it has been proven in \([82]\) that \( Q_n^3 \) is edge-pancyclic. It was posed as an open problem in \([163]\) as to whether their results on bipanconnectivity and bipancyclicity for \( Q_k^2 \) could be extended to \( Q_n^k \), for arbitrary \( n \), and it was posed as an open problem in \([82]\) as to
whether their results on panconnectivity and pancyclicity could be extended to \( Q_n^k \), for arbitrary \( k \). In this chapter, we provide precise answers to both these questions. In addition, we show that when \( k \) is odd, \( Q_n^k \) is \( m \)-panconnected, for \( m = \frac{n(k-1)+2k-6}{2} \), and \((k-1)\)-pancyclic (these bounds are optimal). We also strengthen the results in \([82, 163]\) by introducing a path-shortening technique, called progressive shortening, and show that the construction of paths using this technique enables us to efficiently construct paths in a distributed fashion and so solve a problem relating to the distributed simulation of linear arrays and cycles in a parallel machine whose interconnection network is \( Q_n^k \), even in the presence of a faulty processor (even in \( Q_2^k \), the solution to this problem is not possible using the paths constructed in \([163]\)).

Many structural properties of \( k \)-ary \( n \)-cubes are known, but of particular relevance for us is that a \( k \)-ary \( n \)-cube is vertex-symmetric. Throughout, we assume that addition on tuple elements is modulo \( k \).

It is proven in \([163]\) that \( Q_2^k \) is bipanconnected and (edge-) bipancyclic; however, as to whether \( Q_n^k \), for \( n \geq 3 \), is bipanconnected or bipancyclic was left as an open question. However, in relation to this question, it was proven in \([82]\) that \( Q_n^3 \) is edge-pancyclic, for all \( n \geq 2 \).

Let \( u \) and \( v \) be distinct vertices of \( Q_n^k \) and let \( \rho \) be a path joining \( u \) to \( v \) of length \( m \), where \( m = d(u,v) \) is even. Suppose that there are paths \( \rho_{d(u,v)}, \rho_{d(u,v)+2}, \ldots, \rho_m = \rho \) such that:

- the path \( \rho_i \) joins \( u \) and \( v \) and is of length \( i \), for each \( i = d(u,v), d(u,v)+2, \ldots, m \)
- for each \( i = d(u,v), d(u,v)+2, \ldots, m-2 \), the path \( \rho_{i+2} \) is of the form

\[
U = U_0, U_1, \ldots, U_j, U_j+3, U_j+4, \ldots, U_{i+2} = V
\]

with \( \rho_i \) of the form

\[
U = U_0, U_1, \ldots, U_j, U_{j+3}, U_{j+4}, \ldots, U_{i+2} = V,
\]

for some \( j \in \{0, 1, \ldots, i-1\} \).

Then we say that \( \rho \) can be progressively shortened to obtain paths of all lengths from \( \{l : l = d(u,v), d(u,v)+2, \ldots, m \} \). As we shall see, it will be crucial that our paths can be progressively shortened.
4.2 Existing bipanconnectivity results

In the next section, we improve the constructions from [163] in $Q^k_2$. In Section 3, we look at the general case when $k$ is even, and in Section 4 when $k$ is odd. We outline our application in Section 5 before presenting our conclusions in Section 6.

4.2 Existing bipanconnectivity results

The result from [163] that $Q^k_2$ is bipanconnected (irrespective of whether $k$ is odd or even) is important to our forthcoming results (as the base case of inductions). However, we need to refine the proof from [163] that $Q^k_2$ is bipanconnected in order to obtain a stronger result, involving progressive shortening, and so that we can apply this stronger result later. We remark that it is also crucial that any residual vertex is as stated in Proposition 4.2.1. Our stronger result is as follows.

**Proposition 4.2.1** Let $k \geq 3$ and let $u$ and $v$ be distinct vertices of $Q^k_2$.

1. If $k + d(u,v)$ is odd then there exists a hamiltonian path joining $u$ and $v$ such that this path can be progressively shortened to obtain paths of all lengths from $\{d(u,v) + 2i : 0 \leq i \leq \left\lfloor \frac{k^2 - 1 - d(u,v)}{2} \right\rfloor \}$.

2. If $k + d(u,v)$ is even then there exists an almost-hamiltonian path joining $u$ and $v$ such that the residual vertex is adjacent to $v$ and such that this path can be progressively shortened to obtain paths of all lengths from $\{d(u,v) + 2i : 0 \leq i \leq \left\lfloor \frac{k^2 - 2 - d(u,v)}{2} \right\rfloor \}$.

In particular, $Q^k_2$ is bipanconnected.

Before we prove Proposition 4.2.1, let us illustrate why the proof from [163] that $Q^k_2$ is panconnected will not suffice. Consider Case (a) of Fig. 2 in [163] (in this case, $k$ is even). We have reproduced this figure in Fig. 4.1(a). The authors claim (in a statement prior to Theorem 3) that the almost-hamiltonian path joining $u$ and $v$ can be shortened to a path of length $d(u,v)$ so that paths of lengths $d(u,v), d(u,v) + 2, \ldots, k^2 - 2$ are obtained, and this is indeed the case. However, regard the path from $u$ to $v$ as a curve on the plane and close this curve as shown in Fig. 4.1 with the dotted line. No matter how we progressively shorten the almost-hamiltonian path,
4.3. The general case when \( k \) is even

We begin by examining whether \( Q_n^k \) is bipanconnected or not when \( k \) is even (we reiterate that \( Q_n^k \) is bipartite when \( k \) is even). As remarked earlier, this question was posed as an open problem by Wang, An, Pan, Wang and Qu in [163]. We answer this question precisely.

**Theorem 4.3.1** Let \( k \geq 4 \) and \( n \geq 2 \), with \( k \) even, and let \( u \) and \( v \) be distinct vertices of \( Q_n^k \).

1. If \( d(u,v) \) is odd then there exists a hamiltonian path joining \( u \) and \( v \) such that this path can be progressively shortened to obtain paths of all odd lengths between \( d(u,v) \) and \( k^n - 1 \), inclusive.
4.3. The general case when \( k \) is even

2. If \( d(u, v) \) is even then there exists an almost-hamiltonian path joining \( u \) and \( v \) such that the residual vertex is adjacent to \( v \) and such that this path can be progressively shortened to obtain paths of all even lengths between \( d(u, v) \) and \( k^n - 2 \), inclusive.

In particular, \( Q_n^k \) is bipannconnected.

Proof: The vertex-symmetry of \( Q_n^k \) means that, w.l.o.g., we may suppose that \( u = (0, 0, \ldots, 0) \) and \( v = (v_n, v_{n-1}, v_{n-2}, \ldots, v_1) \), where \( v_i \leq \frac{k}{2} \), for \( i = 1, 2, \ldots, n \), and where \( v \neq (v_n, 0, \ldots, 0) \). For brevity, denote \( v_n \) as \( a \).

Let \( u^i = (i, 0, 0, \ldots, 0) \), for \( 0 \leq i \leq k-1 \); hence, \( u = u^0 \) and \( v \neq u^a \). Partition \( Q_n^k \) over dimension \( n \) to obtain \( Q_n^k(0), Q_n^k(1), \ldots, Q_n^k(k-1) \). We proceed by induction on \( n \). There are two cases: \( d(u^a, v) \) is odd; and \( d(u^a, v) \) is even.

Case (i) \( d(u^a, v) \) is odd.

So, by the induction hypothesis applied to \( Q_n^k(a) \), there exists a hamiltonian path \( \rho_a \) from \( u^a \) to \( v \) in \( Q_n^k(a) \) which can be progressively shortened to obtain paths of
4.3. The general case when $k$ is even

all odd lengths between $d(u^a, v) = d(u, v) - a$ and $k^{n-1} - 1$, inclusive. Note that if the parity of $v$ is even (resp. odd) then $a$ is odd (resp. even).

Denote the vertex $(i, v_{n-1}, v_{n-2}, \ldots, v_1)$ as $v^i$, for $i \in \{0, 1, \ldots, k - 1\}$; so, $v = v^a$.

For each $i \in \{0, 1, \ldots, k - 1\} \setminus \{a\}$, let $\rho_i \in Q_n^k(i)$ be obtained from $\rho_a$ by setting the first component of every vertex of $\rho_a$ at $i$. Note that corresponding vertices of the paths $\rho_0, \rho_1, \ldots, \rho_{k-1}$ induce cycles of length $k$ in $Q_n^k$, e.g., $u^0, u^1, \ldots, u^{k-1}, u^0$ is a cycle of length $k$, as is $v^0, v^1, \ldots, v^{k-1}, v^0$. In particular, the edges of these induced cycles and the edges of the paths $\rho_0, \rho_1, \ldots, \rho_{k-1}$ yield a $k \times k^{n-1}$ grid, with rows $1, 2, \ldots, k$ and columns $1, 2, \ldots, m$, where $m = k^{n-1}$, with 'wrap-around' column edges. Refer to the vertices by their row-column co-ordinates in this grid; so, for example, $u$ is the vertex $(1,1)$ and $v$ is the vertex $(a+1,m)$.

Sub-case (i.a) Suppose that $a$ is even (and so $v$ lies on odd row $a+1$). Consider the path $\rho$ from $u$ to $v$ defined as:

$$(1,1), (2,1), \ldots, (k,1), (k,2), (k-1,2), \ldots, (1,2), (1,3), (2,3), \ldots, (k,3),$$

$$(k,4), (k-1,4), \ldots, (1,4), \ldots, (1,m-3), (2,m-3), \ldots, (k,m-3),$$

$$(k,m-2), (k-1,m-2), \ldots, (1,m-2), (1,m-1), (k,m-1), (k-1,m-1),$$

$$(a+2,m-1), (a+2,m), (a+3,m), \ldots, (k-1,m), (k,m),$$

$$(1,m), (2,m), (2,m-1), (3,m-1), (3,m), (4,m), (4,m-1), \ldots, (a,m), (a,m-1), (a+1,m-1), (a+1,m).$$

The path $\rho$ is hamiltonian and can be visualized as in Fig. 4.3(a). Furthermore, it can trivially be progressively shortened to obtain paths of all odd lengths between $k^{n-1} - 1 + a$ and $k^n - 1$ (inclusive), and so that the path of length $k^{n-1} - 1 + a$ is the path $\rho_0$ in $Q_n^k(0)$, from $u$ to $v^0$, extended with the path in column $m$ of length $a$ to vertex $v$. By above, the path $\rho^0$ can be progressively shortened to obtain paths of all odd lengths between $d(u,v^0) = d(u,v) - a$ and $k^{n-1} - 1$; the result follows.

Sub-case (i.b) Suppose that $a$ is odd (and so $v$ lies on even row $a+1 \geq 2$). Consider the path $\rho$ from $u$ to $v$ defined as:

$$(1,1), (2,1), \ldots, (k,1), (k,2), (k-1,2), \ldots, (1,2), (1,3), (2,3), \ldots, (k,3),$$

$$(k,4), (k-1,4), \ldots, (1,4), \ldots, (1,m-3), (2,m-3), \ldots, (k,m-3),$$
4.3. The general case when \( k \) is even

\[(k, m - 2), (k - 1, m - 2), \ldots, (1, m - 2), (1, m - 1), (k, m - 1), (k - 1, m - 1), \ldots, (a + 2, m - 1), (a + 2, m), (a + 3, m), \ldots, (k - 1, m), (k, m), (1, m), (2, m), (2, m - 1), (3, m - 1), (3, m), (4, m), (4, m - 1), \ldots, (a, m - 1), (a, m), (a + 1, m)\]

(note that the vertex \((a + 1, m - 1)\) does not appear on \(\rho\)).

The path \(\rho\) is almost-hamiltonian and can be visualized as in Fig. 4.3(b). Furthermore, it can trivially be progressively shortened to obtain paths of all even lengths between \(k^{n-1} - 1 + a\) and \(k^n - 2\), and so that the path of length \(k^{n-1} - 1 + a\) is the path \(\rho_0\) in \(Q_n^k(0)\), from \(u\) to \(v^0\), extended with the path in column \(m\) of length \(a\) from \(v^0\) to \(v\). By above, the path \(\rho_0\) can be progressively shortened to obtain paths of all odd lengths between \(d(u, v^0)\) and \(k^{n-1} - 1\). As \(d(u, v) = d(u, v^0) + a\) and the vertex \((a + 1, m - 1)\) is adjacent to \(v\), we obtain the required result.

**Case (ii) \(d(u^a, v)\) is even.**

So, by the induction hypothesis applied to \(Q_n^k(a)\), there exists an almost-hamiltonian path \(\rho_a\) from \(u^a\) to \(v\) in \(Q_n^k(a)\) which can be progressively shortened to obtain paths of all even lengths between \(d(u^a, v)\) and \(k^{n-1} - 2\), and so that the residual vertex of the almost-hamiltonian path \(\rho_a\) is adjacent to \(v\). Note that if the parity of \(v\) is even (resp. odd) then \(a\) is even (resp. odd).

For each \(i \in \{0, 1, \ldots, k - 1\} \setminus \{a\}\), let \(\rho_i \in Q_n^k(i)\) be obtained from \(\rho_a\) by setting the first component of every vertex of \(\rho_a\) at \(i\). As was the case in Case (i), corre-
4.3. The general case when $k$ is even

Corresponding vertices of the paths $\rho_0, \rho_1, \ldots, \rho_{k-1}$ induce cycles of length $k$ in $Q_n^k$. In particular, the edges of these induced cycles and the edges of the paths $\rho_0, \rho_1, \ldots, \rho_{k-1}$ yield a $k \times (k^{n-1} - 1)$ grid, with rows $1, 2, \ldots, k$ and columns $1, 2, \ldots, m - 1$, where $m = k^{n-1}$, with `wrap-around' column edges. Furthermore, if we denote the residual vertex of $\rho_i$ in $Q_n^k(i)$ by $r^i$ then there is an edge $(v^i, r^i)$ in $Q_n^k$, for $i = 0, 1, \ldots, k - 1$; moreover, $r^0, r^1, \ldots, r^{k-1}, r^0$ is a cycle (this is why we focus on the adjacency relationship between the residual vertex and the vertex $v$, as in the statement of the result). Thus, we have a $k \times m$ grid with `wrap-around' column edges, just as we had in Case (i); as before, we refer to the vertices as row-column pairs.

Sub-case (ii.a) Suppose that $a$ is even (and so $v$ lies on odd row $a + 1 \geq 1$ and on column $m - 1$). Consider the path $\rho$ from $u$ to $v$ defined as:

$$
(1, 1), (2, 1), \ldots, (k, 1), (k, 2), (k - 1, 2), \ldots, (1, 2), (1, 3), (2, 3), \ldots,
$$

$$
(k, 3), (k, 4), (k - 1, 4), \ldots, (1, 4), \ldots, (1, m - 3), (2, m - 3), \ldots,
$$

$$
(k, m - 3), (k, m - 2), (k, m - 1), (k, m), (k - 1, m), \ldots, (a + 2, m),
$$

$$
(a + 2, m - 1), (a + 3, m - 1), \ldots, (k - 1, m - 1), (k - 1, m - 2),
$$

$$
(k - 2, m - 2), \ldots, (1, m - 2), (1, m - 1), (1, m), (2, m), (2, m - 1),
$$

$$
(3, m - 1), (3, m), (4, m), (4, m - 1), \ldots, (a, m), (a, m - 1), (a + 1, m - 1)
$$

(note that the vertex $(a + 1, m)$ does not appear on $\rho$). The path $\rho$ is almost-hamiltonian and can be visualized as in Fig. 4.4(a). Furthermore, it can trivially be progressively shortened to obtain paths of all even lengths between $k^{n-1} - 2 + a$ and $k^n - 2$, and so that the path of length $k^{n-1} - 2 + a$ is the path $\rho_0$ in $Q_n^k(0)$, from $u$ to $v^0$, extended with the path in column $m - 1$ of length $a$ from $v^0$ to $v$. By above, the path $\rho^0$ can be progressively shortened to obtain paths of all even lengths between $d(u, v^0)$ and $k^n - 2$. As $d(u, v) = d(u, v^0) + a$ and the vertex $(a + 1, m)$ is adjacent to $v$, we obtain the required result.

Sub-case (ii.b) Suppose that $a$ is odd (and so $v$ lies on even row $a + 1 \geq 2$ and on column $m - 1$). Consider the path $\rho$ from $u$ to $v$ defined as:

$$
(1, 1), (2, 1), \ldots, (k, 1), (k, 2), (k - 1, 2), \ldots, (1, 2), (1, 3), (2, 3), \ldots,
$$

$$
(k, 3), (k, 4), (k - 1, 4), \ldots, (1, 4), \ldots, (1, m - 3), (2, m - 3), \ldots,
$$
4.3. The general case when \( k \) is even

The general case when \( k \) is even

\[
(k, m - 3), (k, m - 2), (k - 1, m - 2), \ldots, (1, m - 2), (1, m - 1),
\]
\[
(1, m), (2, m), (2, m - 1), (3, m - 1), (3, m), (4, m), (4, m - 1), \ldots,
\]
\[
(a, m - 1), (a, m), (a + 1, m), (a + 2, m), \ldots, (k - 1, m), (k, m),
\]
\[
(k, m - 1), (k - 1, m - 1), \ldots, (a + 2, m - 1), (a + 1, m - 1).
\]

The path \( \rho \) is hamiltonian and can be visualized as in Fig. 4.4(b). Furthermore, it can trivially be progressively shortened to obtain paths of all odd lengths between \( k^{n-1} - 2 + a \) and \( k^n - 1 \), and so that the path of length \( k^{n-1} - 2 + a \) is the path \( \rho_0 \) in \( Q_n^k(0) \), from \( u \) to \( v^0 \), extended with the path in column \( m - 1 \) of length \( a \) from \( v^0 \) to \( v \). By above, the path \( \rho^0 \) can be progressively shortened to obtain paths of all even lengths between \( d(u, v^0) = d(u, v) - a \) and \( k^{n-1} - 2 \); thus, we obtain the required result.

All that remains is to deal with the base case of the induction. However, the base case is handled by Proposition 4.2.1.

The following is an immediate corollary of Theorem 4.3.1.

**Corollary 4.3.2** Let \( k \geq 4 \) and \( n \geq 2 \), with \( k \) even. \( Q_n^k \) is edge-bipancyclic.
4.4 The general case when $k$ is odd

We now examine whether $Q_n^k$ is bipanconnected when $k$ is odd. As remarked earlier, this question was posed as an open problem by Wang, An, Pan, Wang and Qu in [163]. We answer this question precisely; in fact, we prove even more as we shall see later.

**Theorem 4.4.1** Let $k \geq 3$ and $n \geq 2$, with $k$ odd, and let $u$ and $v$ be distinct vertices of $Q_n^k$.

1. If $d(u, v)$ is even then there exists a hamiltonian path joining $u$ and $v$ such that this path can be progressively shortened to obtain paths of all even lengths between $d(u, v)$ and $k^n - 1$, inclusive.

2. If $d(u, v)$ is odd then there exists an almost-hamiltonian path joining $u$ and $v$ such that the residual vertex is adjacent to $v$ and such that this path can be progressively shortened to obtain paths of all odd lengths between $d(u, v)$ and $k^n - 2$, inclusive.

In particular, $Q_n^k$ is bipan connected.

**Proof:** The proof is very similar in structure to that of Theorem 4.3.1 and we adopt the exact same notation as in that proof. Again, we proceed by induction on $n$ and there are two cases, according to whether $d(u^a, v)$ is odd or even.

**Case (i) $d(u^a, v)$ is even.**

So, by the induction hypothesis, there exists a hamiltonian path $\rho_a$ from $u^a$ to $v$ in $Q_n^k(a)$ which can be progressively shortened to obtain paths of all even lengths between $d(u^a, v) = d(u, v) - a$ and $k^{n-1} - 1$, inclusive. As in the proof Theorem 4.3.1, the paths $\rho_0, \rho_1, \ldots, \rho_{k-1}$ yield a $k \times k^{n-1}$ grid, with rows $1, 2, \ldots, k$ and columns $1, 2, \ldots, m$, where $m = k^{n-1}$, with 'wrap-around' column edges.

**Sub-case (i.a) Suppose that $a$ is even (and so $v$ lies on odd row $a + 1 \geq 1$ and on column $m$).** Consider the path $\rho$ from $u$ to $v$ defined as:

$$(1, 1), (2, 1), \ldots, (k, 1), (k, 2), (k - 1, 2), \ldots, (1, 2), (1, 3), (2, 3), \ldots, (k, 3),

(k, 4), (k - 1, 4), \ldots, (1, 4), \ldots, (k, m - 3), (k - 1, m - 3), \ldots, (1, m - 3),$$
4.4. The general case when $k$ is odd

Figure 4.5: The different cases when $d(u^a, v)$ is even.

$(1, m - 2), (2, m - 2), \ldots, (k, m - 2), (k, m - 1), (k, m), (k - 1, m),$

$(k - 1, m - 1), (k - 2, m - 1), (k - 2, m), \ldots, (a + 2, m), (a + 2, m - 1),$

$(a + 1, m - 1), (a, m - 1), \ldots, (1, m - 1), (1, m), (2, m), \ldots, (a + 1, m).$

The path $\rho$ is hamiltonian and can be visualized as in Fig. 4.5(a). Similarly to as in the proof of Theorem 4.3.1, $\rho$ can be progressively shortened to obtain paths of all even lengths between $d(u, v)$ and $k^n - 1$.

Sub-case (i.b) Suppose that $a$ is odd (and so $v$ lies on even row $a + 1 \geq 2$ and on column $m$). Consider the path $\rho$ from $u$ to $v$ defined as:

$(1, 1), (2, 1), \ldots, (k, 1), (k, 2), (k - 1, 2), \ldots, (1, 2), (1, 3), (2, 3), \ldots, (k, 3),$

$(k, 4), (k - 1, 4), \ldots, (1, 4), \ldots, (k, m - 3), (k - 1, m - 3), \ldots, (1, m - 3),$

$(1, m - 2), (2, m - 2), \ldots, (k, m - 2), (k, m - 1), (k, m), (k - 1, m),$

$(k - 1, m - 1), (k - 2, m - 1), (k - 2, m), (k - 3, m), (k - 3, m - 1), \ldots,$

$(a + 2, m - 1), (a + 1, m - 1), (a, m - 1), \ldots, (1, m - 1), (1, m),$

$(2, m), \ldots, (a + 1, m)$

(note that the vertex $(a + 2, m)$ does not appear on $\rho$). The path $\rho$ is almost-hamiltonian and can be visualized as in Fig. 4.5(b). Similarly to as in the proof of Theorem 4.3.1, $\rho$ can be progressively shortened to obtain paths of all odd lengths between $d(u, v)$ and $k^n - 2$. 

4.4. The general case when $k$ is odd

Case (ii) $d(u^a, v)$ is odd.

So, by the induction hypothesis, there exists an almost-hamiltonian path $\rho_a$ from $u^a$ to $v$ in $Q_k^a(a)$ which can be progressively shortened to obtain paths of all odd lengths between $d(u^a, v) = d(u, v) - a$ and $k^n - 2$, and so that the residual vertex of the almost-hamiltonian path $\rho_a$ is adjacent to $v$. As in the proof Theorem 4.3.1, the paths $\rho_0, \rho_1, \ldots, \rho_{k-1}$ and the residual vertices yield a $k \times k^{n-1}$ grid, with rows $1, 2, \ldots, k$ and columns $1, 2, \ldots, m$, where $m = k^{n-1}$, with 'wrap-around' column edges.

Sub-case (ii.a) Suppose that $a$ is odd (and so $v$ lies on even row $a + 1 > 2$ and on column $m - 1$). Consider the path $\rho$ from $u$ to $v$ defined as:

$$(1, 1), (2, 1), \ldots, (k, 1), (k, 2), (k - 1, 2), \ldots, (1, 2), (1, 3), (2, 3), \ldots, (k, 3),$$

$$(k, 4), (k - 1, 4), \ldots, (1, 4), \ldots, (k, m - 3), (k, m - 2), (k - 1, m - 2),$$

$$(k - 1, m - 3), \ldots, (a + 2, m - 3), (a + 2, m - 2), (a + 1, m - 2),$$

$$(a + 1, m - 3), (a, m - 3), (a, m - 2), \ldots, (4, m - 2), (4, m - 3),$$

$$(3, m - 3), (3, m - 2), (2, m - 2), (2, m - 3), (1, m - 3), (1, m - 2),$$

$$(1, m - 1), (k, m - 1), (k - 1, m - 1), \ldots, (a + 2, m - 1), (a + 2, m),$$

$$(a + 3, m), \ldots, (k, m), (1, m), (2, m), (2, m - 1), (3, m - 1), (3, m),$$

$$(4, m), (4, m - 1), \ldots, (a, m - 1), (a, m), (a + 1, m), (a + 1, m - 1).$$

The path $\rho$ is hamiltonian and can be visualized as in Fig. 4.6(a). Similarly to as in the proof of Theorem 4.3.1, $\rho$ can be progressively shortened to obtain paths of all even lengths between $d(u, v)$ and $k^n - 1$.

Sub-case (ii.b) Suppose that $a$ is even (and so $v$ lies on odd row $a + 1 \geq 1$ and on column $m - 1$). Consider the path $\rho$ from $u$ to $v$ defined as:

$$(1, 1), (2, 1), \ldots, (k, 1), (k, 2), (k - 1, 2), \ldots, (1, 2), (1, 3), (2, 3), \ldots, (k, 3),$$

$$(k, 4), (k - 1, 4), \ldots, (1, 4), \ldots, (k, m - 3), (k - 1, m - 3), \ldots, (1, m - 3),$$

$$(1, m - 2), (1, m - 1), (1, m), (2, m), (2, m - 1), (2, m - 2), (3, m - 2),$$

$$(3, m - 1), (3, m), (4, m), (4, m - 1), (4, m - 2), \ldots, (a, m), (a, m - 1),$$

$$(a, m - 2), (a + 1, m - 2), (a + 2, m - 2), \ldots, ((k, m - 2), k, m - 1),$$

$$(k, m), (k, m - 1), \ldots, (a, m - 1), (a, m), (a + 1, m), (a + 1, m - 1).$$
4.4. The general case when \( k \) is odd

\[ (k, m), (k - 1, m), (k - 1, m - 1), (k - 2, m - 1), \ldots, (a + 2, m), \]
\[ (a + 2, m - 1), (a + 1, m - 1) \]

(note that the vertex \((a + 1, m)\) does not appear on \( \rho \)). The path \( \rho \) is almost-hamiltonian and can be visualized as in Fig. 4.6(b). Similarly to as in the proof of Theorem 4.3.1, \( \rho \) can be progressively shortened to obtain paths of all odd lengths between \( d(u, v) \) and \( k^n - 2 \).

However, the base case is handled by Proposition 4.2.1.

The following is an immediate corollary of Theorem 4.4.1.

**Corollary 4.4.2** Let \( k \geq 3 \) and \( n \geq 2 \), with \( k \) odd. \( Q_n^k \) is edge-bipancyclic.

As remarked earlier, bipanconnectivity and bipancyclicity are concepts which make most sense in the context of bipartite graphs, such as the graphs \( Q_n^k \), for \( k \) even. However, when \( k \) is odd, \( Q_n^k \) is not bipartite and it is possible that odd cycles might exist, as well as odd and even length paths between vertices \( u \) and \( v \). As we shall see, this is indeed the case but not universally.

Henceforth, \( k \) is odd. Consider the vertices \( u = (0, 0, \ldots, 0) \) and \( v = (v_n, v_{n-1}, \ldots, v_1) \) of \( Q_n^k \), where (as usual) we assume w.l.o.g. that \( v_i \leq \frac{k - 1}{2} \), for \( i = 1, 2, \ldots, n \). Consider any path from \( u \) to \( v \) that does not use any 'wrap-around' edge, i.e., an edge where the \( i \)th component of one incident vertex is \( k - 1 \) and where the \( i \)th component of the other incident vertex is 0, for some \( i \). Such a path must alternate
4.4. The general case when $k$ is odd

between odd parity and even parity vertices; thus, such paths are either all of even length or all of odd length (depending upon whether $d(u, v)$ is even or odd). Suppose that $d(u, v)$ is odd (and so all such paths are of odd length). Let $i$ be such that $v_i$ is maximal from amongst $\{v_n, v_{n-1}, \ldots, v_1\}$. Any path from $u$ to $v$ of length at most

$$v_n + \ldots + v_{i+1} + (k - v_i - 1) + v_{i-1} + \ldots + v_1 = d(u, v) + k - 2v_i - 1$$

cannot use a wrap-around edge and so must be of odd length. Consequently, there are no even length paths from $u$ to $v$ of length less than $d(u, v) + k - 2v_i$. Identical reasoning implies that if $d(u, v)$ is even then there are no odd length paths from $u$ to $v$ of length less than $d(u, v) + k - 2v_i$. Consequently, we have a lower bound on the length of a shortest path, joining $u$ and $v$ and of parity different from that of $d(u, v)$.

Choose the vertex $v$ of $Q^k_n$ to be such that $v_n = 1$ and $v_j = 0$, for $j = 1, 2, \ldots, n - 1$. Thus, there exists a vertex $v$ such that $d(u, v)$ is odd and there are no paths joining $u$ and $v$ of even length less than $d(u, v) + k - 2$. There clearly also exists a vertex $v'$ such that $d(u, v')$ is even and there are no paths joining $u$ and $v'$ of odd length less than $d(u, v) + k - 2$ (for example, choose $v' = (1, 1, 0, \ldots, 0)$). Consequently, as we are interested in general statements concerning all pairs of distinct vertices from $Q^k_n$, we shall only look for even (resp. odd) length paths joining $u$ and $v$ of length at least $d(u, v) + k - 2$, when $d(u, v)$ is odd (resp. even).

**Theorem 4.4.3** Let $k \geq 3$ and $n \geq 2$, with $k$ odd, and let $u$ and $v$ be distinct vertices of $Q^k_n$. There are paths joining $u$ and $v$ of all lengths in $\{i : d(u, v) + k - 3 \leq i \leq k^n - 1\}$. Furthermore, this result is optimal in that there exist distinct vertices $u$ and $v$ of $Q^k_n$ for which $d(u, v)$ is odd (resp. even) and there are no even-length (resp. odd-length) paths joining $u$ and $v$ of length less than $d(u, v) + k - 2$.

**Proof:** The proof is very similar in structure to that of Theorem 4.4.1 and we adopt the exact same notation as in that proof (and in the proof of Theorem 4.3.1). There are two cases, according to whether $d(u^a, v)$ is odd or even. Given the earlier proofs, we are much briefer with our arguments here.

**Case (i)** $d(u^a, v)$ is even.
4.4. The general case when \( k \) is odd

By Theorem 4.4.1, there exists a Hamiltonian path \( \rho_a \) from \( u^a \) to \( v \) in \( Q_n^k(a) \) which can be progressively shortened to obtain paths of all even lengths between \( d(u^a, v) = d(u, v) - a \) and \( k^{n-1} - 1 \), inclusive. As in the proofs of Theorems 4.3.1 and 4.4.1, the paths \( \rho_0, \rho_1, \ldots, \rho_{k-1} \) yield a \( k \times k^{n-1} \) grid, with rows 1,2,\ldots,\( k \) and columns 1,2,\ldots,\( m \), where \( m = k^{n-1} \), with 'wrap-around' column edges.

Sub-case (i.a) Suppose that \( a \) is even (and so \( v \) lies on odd row \( a + 1 \geq 1 \) and on column \( m \)). Build the path \( \rho \) as depicted in Fig. 4.7(a). It is easy to see that \( \rho \) has length \( k^{n} - 2 \) and can be progressively shortened to obtain paths of all odd lengths between \( (k - 1) + d(u^a, v) + a + 1 = d(u, v) + k + k^{n} - 2 \) (shorten so that the resulting sub-path of length \( k^{n-1} - 1 \) lies on row \( k \)).

Sub-case (i.b) Suppose that \( a \) is odd (and so \( v \) lies on even row \( a + 1 \geq 2 \) and on column \( m \)). Build the path \( \rho \) as depicted in Fig. 4.7(b). It is easy to see that \( \rho \) has length \( k^{n} - 1 \) and can be progressively shortened to obtain paths of all even lengths between \( (k - 1) + d(u^a, v) + a + 1 = d(u, v) + k + k^{n} - 1 \).

Case (ii) \( d(u^a, v) \) is odd.

By Theorem 4.4.1, there exists an almost-Hamiltonian path \( \rho_a \) from \( u^a \) to \( v \) in \( Q_n^k(a) \) which can be progressively shortened to obtain paths of all odd lengths between \( d(u^a, v) = d(u, v) - a \) and \( k^{n-1} - 2 \), inclusive, and so that the residual vertex is adjacent to \( v \). As before, the paths \( \rho_0, \rho_1, \ldots, \rho_{k-1} \) and the residual vertices yield a \( k \times k^{n-1} \) grid, with rows 1,2,\ldots,\( k \) and columns 1,2,\ldots,\( m \), where \( m = k^{n-1} \), with
4.4. The general case when $k$ is odd

Sub-case (ii.a) Suppose that $a$ is odd (and so $v$ lies on even row $a + 1 \geq 2$ and on column $m - 1$). Build the path $\rho$ as depicted in Fig. 4.8(a). It is easy to see that $\rho$ has length $k^n - 2$ and can be progressively shortened to obtain paths of all odd lengths between $(k - 1) + d(u^a, v) + a + 1 = d(u, v) + k$ and $k^n - 2$.

Sub-case (ii.b) Suppose that $a$ is even (and so $v$ lies on odd row $a + 1 \geq 1$ and on column $m - 1$). Build the path $\rho$ as depicted in Fig. 4.8(b). It is easy to see that $\rho$ has length $k^n - 1$ and can be progressively shortened to obtain paths of all even lengths between $(k - 1) + d(u^a, v) + a + 1 = d(u, v) + k$ and $k^n - 1$.

In order to complete the construction of our paths, we deal with some special cases. W.l.o.g., assume that $v_n \neq 0$. There is trivially a path of length

$$(k - v_n) + v_{n-1} + \ldots + v_1 = d(u, v) + k - 2v_n \leq d(u, v) + k - 2$$

joining $u$ and $v$. We can easily lengthen this path to obtain a path of length $d(u, v) + k - 2$ joining any distinct vertices $u$ and $v$. Hence, no matter which vertex $v$ is, Theorem 4.4.1 yields paths as in the statement of the result. Optimality follows by the argument presented prior to the statement of the result.

Note that putting $k = 3$ in Theorem 4.4.3 yields the result from [82] that $Q_n^3$ is edge-pancyclic, and also resolves the question for arbitrary $k$, as was posed in [82]. The following corollary is immediate, given the fact that the diameter of $Q_n^k$, when $k$ is odd, is $\frac{n(k-1)}{2}$.
Corollary 4.4.4 Let $k \geq 3$ and $n \geq 2$, with $k$ odd. The $k$-ary $n$-cube $Q^n_k$ is $m$-panconnected, for $m = \frac{n(k-1)+2k-6}{2}$, and $(k-1)$-pancyclic.

As remarked earlier, the bounds in Corollary 4.4.4 are optimal.

4.5 An application

We give here the outline of an application where we require our paths to be progressively shortened and where alternative shortening methods will not suffice.

Consider a parallel machine whose underlying interconnection network is a $k$-ary $n$-cube, and where this machine is required to solve problems specifically designed for a cycle of processors (amongst other problems), with the number of processors involved in the cycle being variable. Moreover, there is known to be a faulty processor in the machine and this faulty processor cannot be used in any embedded cycle. Furthermore, the location of the fault is not known and any cycle must be constructed in a distributed fashion, through message-passing between processors.

For simplicity, suppose that $k$ is even and $n = 2$; consequently, any cycle we construct must have even length. We begin our construction by processor $(0,0)$ attempting to construct a hamiltonian path to processor $(0,1)$ according to the construction in Proposition 4.2.1. Actually, the path is constructed as in Case 1.3 of Theorem 1 of [163]. It is important to note that the constructions in Proposition 4.2.1 (and Theorems 1 and 3 of [163]) are of such a uniform nature that the processor at the head of the path constructed so far can calculate in constant time the name of the next processor on the path, and can send a message to this processor thus extending the path constructed so far. If there were no faults then this construction would terminate with a hamiltonian path from $(0,0)$ to $(0,1)$ laid out in the $k$-ary 2-cube. However, the construction will halt when the faulty processor is encountered (we assume that the processor immediately before the fault on the constructed path can detect that the next processor is faulty).

Let $p$ be the processor that detects that the faulty processor is the next processor on the path, and suppose that this faulty processor is $f = (i,j)$. The processor $p$ sends a message to processor $s = (i+1,j)$ (over at most 4 hops, with addition
4.6. Conclusions

modulo $k$) that it should use the construction of Proposition 4.2.1 to embark on the construction of a path of length $k^2 - 2$ to the processor $(i, j - 1)$. Note that the path, as shown in Fig. 4.2(b) (that is, the amended construction of a case from [163]), avoids the faulty processor $f$. We reiterate that the uniform nature of the construction is such that the processor at the head of the path constructed so far can calculate in constant time the name of the next processor on the path, and can send a message to this processor thus extending the path constructed so far. Having reached the processor $(i, j - 1)$, we actually truncate the path at processor $t = (i + 1, j - 1)$. Thus, we have a path of length $k^2 - 3$ from processor $s$ to $t$, avoiding processor $(i, j - 1)$ and the faulty processor $f$. Moreover, this path can be progressively shortened so as to obtain any odd length path (of length at most $k^2 - 3$) joining $s$ to $s$ (and avoiding $f$). Furthermore, again because of the uniformity of the construction and also the uniformity of the progressive shortening, this progressive shortening can easily be completed by message-passing between the processors. In fact, message-passing can be used so that every processor $q$ on the path computes a list of triples of the form $(q^+, q^-, i)$ detailing that $q$ appears on a path of length $i$ from $s$ to $t$ so that that the processor $q^-$ (resp. $q^+$) is the next processor on this path moving towards $s$ (resp. $t$). The existence of the edge $(s, t)$ gives our embedded fault-avoiding cycles of varying lengths.

The above construction can be generalized to an analogous construction of fault-avoiding paths and cycles in $Q_k^k$ where there is a faulty processor. As we stated above, we have not presented the precise details of this generalization; what suffices is that the general principle has been presented and any interested reader could implement the construction if needs be. We envisage that there are many other applications of progressive shortening but we have chosen not to explore these applications here.

4.6 Conclusions

In tandem with [82,163], we have resolved completely the main questions concerning panconnectivity, bipanconnectivity, pancyclicity and bipancyclicity for a $k$-ary $n$-
4.6. Conclusions

cube $Q^k_n$, when $k \geq 3$ and $n \geq 2$. In doing so, we have introduced the new concept of the progressive shortening of a path and shown how this concept can be used to solve a problem related to the embedding of linear arrays and cycles of processors in a distributed-memory multiprocessor whose interconnection network is a $k$-ary $n$-cube and where there is one faulty processor.

As directions for future research, we would like to see more applications of progressive shortening (and feel that the concept will prove to be more widely applicable). Also, we would like to see results on panconnectivity, pancyclicity, and so forth, extended to $k$-ary $n$-cubes in which there may be (a limited number of) faulty vertices or edges.
Chapter 5

Augmented \( k \)-ary \( n \)-cube

In this chapter, we define an interconnection network \( AQ_{n,k} \) which we call the augmented \( k \)-ary \( n \)-cube by extending a \( k \)-ary \( n \)-cube in a manner analogous to the existing extension of an \( n \)-dimensional hypercube to an \( n \)-dimensional augmented cube. We prove that the augmented \( k \)-ary \( n \)-cube \( AQ_{n,k} \) has a number of attractive properties (in the context of parallel computing). For example, we show that the augmented \( k \)-ary \( n \)-cube \( AQ_{n,k} \) is a Cayley graph (and so is vertex-symmetric); has connectivity \( 4n - 2 \), and is such that we can build a set of \( 4n - 2 \) mutually disjoint paths joining any two distinct vertices so that the path of maximal length has length at most \( \max\{(n - 1)k - (n - 2), k + 7\} \); has diameter \( \left\lceil \frac{k}{3} \right\rceil + \left\lceil \frac{k-1}{3} \right\rceil \), when \( n = 2 \); and has diameter at most \( \frac{k}{3}(n + 1) \), for \( n \geq 3 \) and \( k \) even, and at most \( \frac{k}{4}(n + 1) + \frac{7}{4} \), for \( n \geq 3 \) and \( k \) odd.

5.1 Introduction

Hypercubes are perhaps the most well known of all interconnection networks for parallel computing, given their basic simplicity, their generally desirable topological and algorithmic properties, and the extensive investigation they have undergone (not just in the context of parallel computing but also in discrete mathematics in general; see, for example, [139] for some essential properties of hypercubes). However, a multitude of different interconnection networks have been devised and developed in a continuing search for improved performance, with many of these
networks having hypercubes at their roots. Amongst these generalisations of hypercubes are $k$-ary $n$-cubes [42], augmented cubes [41], cube-connected cycles [132], twisted cubes [75], twisted $n$-cubes [53], crossed cubes [50], folded hypercubes [51], Mcubes [148], Möbius cubes [103], generalised twisted cubes [33], shuffle cubes [112], $k$-skip enhanced cubes [159], twisted hypercubes [99], supercubes [143], and Fibonacci cubes [88].

Perhaps the most popular of these generalisations are the $k$-ary $n$-cubes [42]. Another generalisation of hypercubes are augmented cubes, recently proposed by Choudum and Sunitha [41] as improvements over hypercubes. Hypercubes and augmented cubes (of the same dimensions) have the same sets of vertices. However, whereas the recursive construction of an $n$-dimensional hypercube is to take two copies of an $(n-1)$-dimensional hypercube and join corresponding pairs of vertices, the recursive construction of an $n$-dimensional augmented cube $AQ_n$ is to take two copies of an $(n-1)$-dimensional augmented cube and as well as joining corresponding pairs of vertices, pairs of vertices of Hamming distance $n - 1$ are also joined (that is, vertices that are different in every component). Choudum and Sunitha show that an $n$-dimensional augmented cube $AQ_n$: has $2^n$ vertices and $n2^n$ edges; has diameter $\lceil \frac{n}{2} \rceil$; has connectivity $2n - 1$; is a Cayley graph and so is vertex-symmetric; and has an $O(n)$ time optimal routing algorithm.

In this chapter, and inspired by [41], we extend a $k$-ary $n$-cube in a manner analogous to the extension of an $n$-dimensional hypercube to an $n$-dimensional augmented cube. Our definition of an augmented $k$-ary $n$-cube $AQ_{n,k}$, in comparison with that in [41], is not a straightforward generalisation; however, we believe that it does reflect the essence of the extension in [41], and our structural results bear this out. We give two different definitions of an augmented $k$-ary $n$-cube in Section 5.2 and show that they yield the same interconnection network. In Section 5.3, we show that an augmented $k$-ary $n$-cube $AQ_{n,k}$ is vertex-symmetric and, furthermore, a Cayley graph. In Section 5.4, we show that an augmented $k$-ary $n$-cube $AQ_{n,k}$ has connectivity $4n - 2$, and that we can build a set of $4n - 2$ mutually disjoint paths joining any two distinct vertices so that the path of maximal length has length at most at most $\max\{(n - 1)k - (n - 2), k + 7\}$. In Section 5.5, we examine the
5.2. Basic definitions

We assume throughout that addition on tuple elements is modulo \( k \). Recall the definition of the \( k \)-ary \( n \)-cube \( Q^k_n \): the vertex set \( V(Q^k_n) \) is \{ \( (a_n, a_{n-1}, \ldots, a_1) : 0 < a_i < k - 1 \) \}; and the edge set \( E(Q^k_n) \) is \{ \( (u, v) : \) either \( u_i = v_i - 1 \) or \( u_i = v_i + 1 \), for some \( i \), and \( u_j = v_j \), for all \( i \neq j \) \}. Whilst we regard all graphs defined in this chapter as undirected, our definitions define all edges from the perspective of a given vertex. Thus, in our definition of \( Q^k_n \) we define the (undirected) edge \( (u, v) \) twice: once from the perspective of \( u \), as the edge \( (u, v) \); and once from the perspective of \( v \), as the edge \( (v, u) \). The reason we do this is that later we shall define paths in our graphs and an undirected edge will be regarded differently depending upon the direction it is being traversed in the path. The following definition adheres to this convention.

**Definition 5.2.1** Let \( n \geq 1 \) and \( k \geq 3 \) be integers. The augmented \( k \)-ary \( n \)-cube \( AQ_{n,k} \) has \( k^n \) vertices, each labelled by an \( n \)-bit string \( (a_n, a_{n-1}, \ldots, a_1) \), with \( 0 < a_i < k - 1 \), for \( 1 < i < n \). There is an edge joining vertex \( u = (u_n, u_{n-1}, \ldots, u_1) \) to vertex \( v = (v_n, v_{n-1}, \ldots, v_1) \) if, and only if:

- \( v_i = u_i - 1 \) (resp. \( v_i = u_i + 1 \)), for some \( 1 \leq i \leq n \), and \( v_j = u_j \), for all \( 1 \leq j \leq n, j \neq i \); call the edge \( (u, v) \) an \((i, -1)\)-edge (resp. an \((i, +1)\)-edge); or

- for some \( 2 \leq i \leq n \), \( v_i = u_i - 1, v_{i-1} = u_{i-1} - 1, \ldots, v_1 = u_1 - 1 \) (resp. \( v_i = u_i + 1, v_{i-1} = u_{i-1} + 1, \ldots, v_1 = u_1 + 1 \)), \( v_j = u_j \), for all \( j > i \); call the edge \( (u, v) \) a \((\leq i, -1)\)-edge (resp. a \((\leq i, +1)\)-edge).

We emphasise that the graph \( AQ_{n,k} \) is undirected but that edges are labelled differently, as an \((i, +1)\)-edge or as an \((i, -1)\)-edge, for example, according to the perceived
The augmented \( k \)-ary \( n \)-cube \( AQ_{n,k} \) can also be recursively defined as follows (the proof of this fact is a simple induction).

**Definition 5.2.2** Fix \( k \geq 3 \). The augmented \( k \)-ary \( 1 \)-cube \( AQ_{1,k} \) has vertex set \( \{0, 1, \ldots, k - 1\} \) and there is an edge joining vertex \( u \) to vertex \( v \) if, and only if, \( v = u + 1 \) or \( v = u - 1 \). Fix \( n \geq 2 \). Take \( k \) copies of an augmented \( k \)-ary \( (n - 1) \)-cube \( AQ_{n-1,k} \) and for the \( i \)th copy, add an extra number \( i \) as the \( n \)th bit of each vertex (all vertices have the same \( n \)th bit if they are in the same augmented \( k \)-ary \( (n - 1) \)-cube). Four more edges are added for each vertex, namely the \( (n, -1) \)-edge, the \( (n, +1) \)-edge, the \( (\leq n, -1) \)-edge and the \( (\leq n, +1) \)-edge (as defined in Definition 5.2.1).

With respect to the above definition, we refer to the subgraph of \( AQ_{n,k} \) induced by the vertices whose first component is \( i \), for some fixed \( i \in \{0, 1, \ldots, k - 1\} \), as \( AQ_{n-1,k}^i \) (this subgraph is clearly a copy of \( AQ_{n-1,k} \)).

Clearly (from the definition of \( AQ_{n,k} \)), when \( n \geq 2 \), \( AQ_{n,k} \) has \( n^k \) vertices, 
\((2n - 1)n^k\) edges, and every vertex has degree \( 4n - 2 \).

We adopt the following notation with regard to identifying specific vertices relevant to a given vertex in \( AQ_{n,k} \). Let \( v = (v_n, v_{n-1}, \ldots, v_1) \) be some vertex of \( AQ_{n,k} \). For each \( i \in \{0, 1, \ldots, k - 1\} \) and each \( j \in \{1, 2, \ldots, n\} \), we denote the vertex \((v_n, v_{n-1}, \ldots, v_{j+1}, i, v_{j-1}, \ldots, v_1)\) by \( v_{j}^i \). For \( j \in \{1, 2, \ldots, n\} \), we refer to the neighbour \((v_n, \ldots, v_{j+1}, v_j + 1, v_{j-1}, \ldots, v_1)\) (resp. \((v_n, \ldots, v_{j+1}, v_j - 1, v_{j-1}, \ldots, v_1)\), \((v_n, \ldots, v_{j+1}, v_j + 1, v_{j-1} + 1, \ldots, v_1 + 1)\), \((v_n, \ldots, v_{j+1}, v_j - 1, v_{j-1} - 1, \ldots, v_1 - 1)\)) as \( v_{(j,+1)} \) (resp. \( v_{(j,-1)} \), \( v_{(\leq j,+1)} \), \( v_{(\leq j,-1)} \)). We can combine our notation as the following example shows: \( v_{(j,+1)}^{(j+1)} \) denotes the vertex obtained by taking the vertex \( v_{(j,+1)} \) and fixing its \( n \)th component at 3 whilst leaving all other components as they were.

Paths in graphs are given as sequences of vertices (on occasion, a path might consist of a solitary vertex). A path in \( AQ_{n,k} \) might be specified by the source vertex and a sequence of labels detailing the edges to be traversed, *e.g.*, the path in \( AQ_{3,5} \) detailed as having the source vertex \((0, 0, 0)\) and then following the edges labelled \((\leq 2, +1), (3, -1), (1, +1)\) is actually the path \((0, 0, 0), (0, 1, 1), (4, 1, 1), (4, 1, 2)\).
5.3. Symmetry

The augmented 5-ary 2-cube is depicted in Fig. 5.1 where the edges of the underlying 5-ary 2-cube (that is, the (2, +1)-edges, the (2, −1)-edges, the (1, +1)-edges and the (1, −1)-edges) are drawn using narrow pen and the “augmented” edges (that is, the (≤ 2, +1)-edges and the (≤ 2, −1)-edges) are drawn using broad pen.

5.3 Symmetry

In this section, we examine $AQ_{n,k}$ as to any symmetric properties it might have. We begin with a useful lemma which will be used to reduce case analyses in subsequent proofs, and the proof of which is trivial.

**Lemma 5.3.1**  
(a) The following are automorphisms of $AQ_{n,k}$:

(i) the mapping taking the vertex $(v_n, v_{n-1}, \ldots, v_1)$ to $(v_n - a_n, v_{n-1} - a_{n-1}, \ldots, v_1 - a_1)$, where $(a_n, a_{n-1}, \ldots, a_1) \in \{0, 1, \ldots, k-1\}^n$ is fixed;

(ii) the mapping taking the vertex $(v_n, v_{n-1}, \ldots, v_1)$ to $(\epsilon v_n, \epsilon v_{n-1}, \ldots, \epsilon v_1)$, where $\epsilon \in \{+1, -1\}$ is fixed.

(b) For $i, j \in \{0, 1, \ldots, k-1\}$, the mapping taking the vertex $(i, v_{n-1}, v_{n-2}, \ldots, v_1)$ to $(j, v_{n-1}, v_{n-2}, \ldots, v_1)$ is an isomorphism of $AQ_{n-1,k}^i$ to $AQ_{n-1,k}^j$.

(c) The mapping taking the vertex $(u, v)$ to the vertex $(v, u)$ is an automorphism of $AQ_{2,k}$.
The property of a graph being vertex-symmetric is important when that graph is used as an interconnection network for parallel computing, for having a vertex-symmetric interconnection network makes parallel algorithm design and topological analysis easier, as well as allowing flexibility in, for example, linear array simulations.

An immediate corollary of Lemma 5.3.1 is the following.

**Corollary 5.3.2** The augmented \( k \)-ary \( n \)-cube \( AQ_{n,k} \) is vertex-symmetric.

**Proof:** Given vertices \( u = (u_n, u_{n-1}, \ldots, u_1) \) and \( v = (v_n, v_{n-1}, \ldots, v_1) \) of \( AQ_{n,k} \), by Lemma 5.3.1, the mapping taking an arbitrary vertex \( (w_n, w_{n-1}, \ldots, w_1) \) to \( (w_n - (u_n - v_n), w_{n-1} - (u_{n-1} - v_{n-1}), \ldots, w_1 - (u_1 - v_1)) \) is an automorphism mapping \( u \) to \( v \).

However, we can do better. Let \( \Gamma \) be a finite group and let \( S \subseteq \Gamma \) be a set of generators of \( \Gamma \) not containing the identity and closed under inversion; that is, \( s^{-1} \in S \) whenever \( s \in S \). The simple undirected graph \( G(\Gamma, S) \) with vertex set \( \Gamma \) and where two vertices \( g \) and \( h \) are adjacent if, and only if, \( gh^{-1} \in S \), is called the *Cayley graph* of \( \Gamma \) (with generating set \( S \)). Knowledge that an interconnection network is a Cayley graph not only immediately yields that the graph is vertex-symmetric but also provides an algebraic description of the graph that will be useful in, for example, developing routing algorithms.

Let \( (Z_k)^n \) denote the \( n \)-fold Cartesian product of the group \( (Z_k, \oplus_k) \), where \( Z_k = \{0, 1, \ldots, k - 1\} \) and where \( \oplus_k \) denotes addition modulo \( k \). Let \( x = (x_n, x_{n-1}, \ldots, x_1) \in (Z_k)^n \); so \( x^{-1} = (k - x_n, k - x_{n-1}, \ldots, k - x_1) \).

**Proposition 5.3.1** For every \( n \geq 1 \), \( AQ_{n,k} \cong G((Z_k)^n, S) \), where \( S \) is the set:

\[
\{(0, \ldots, 0, 0, k - 1, k - 1), (0, \ldots, 0, k - 1, k - 1, k - 1), \ldots,
(k - 1, \ldots, k - 1, k - 1), (0, \ldots, 0, 0, 1, 1), (0, \ldots, 0, 1, 1, 1), \ldots, (1, \ldots, 1, 1),
(k - 1, 0, 0, \ldots, 0), (0, k - 1, 0, \ldots, 0), \ldots, (0, \ldots, 0, k - 1),
(1, 0, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)\}.
\]

**Proof:** By definition, \( V(AQ_{n,k}) = Z_k \times Z_k \times \ldots \times Z_k \) (repeated \( n \) times). Let \( u = (u_n, u_{n-1}, \ldots, u_1) \) and \( v = (v_n, v_{n-1}, \ldots, v_1) \) be vertices of \( AQ_{n,k} \).
Suppose that $u$ and $v$ are adjacent in $AQ_{n,k}$. So, for some $i$, one of the following holds:

1. $v = (u_n, u_{n-1}, ..., u_{i+1}, u_i \oplus_k 1, u_{i-1}, ..., u_1)$
2. $v = (u_n, u_{n-1}, ..., u_{i+1}, u_i \oplus_k 1, u_{i-1} \oplus_k 1, ..., u_1 \oplus_k 1)$
3. $v = (u_n, u_{n-1}, ..., u_{i+1}, u_i \oplus_k (k-1), u_{i-1}, ..., u_1)$
4. $v = (u_n, u_{n-1}, ..., u_{i+1}, u_i \oplus_k (k-1), u_{i-1} \oplus_k (k-1), ..., u_1 \oplus_k (k-1))$

Thus, we have (respectively):

1. $u \oplus_k v^{-1} = (u_n \oplus_k (k - u_{n-1}), ..., u_{i+1} \oplus_k (k - u_{i+1}), u_i \oplus_k (k - (u_i + 1)),$
   
   \[ u_{i+1} \oplus_k (k - u_{i+1}), ..., u_0 \oplus_k (k - u_0) \]
   
   \[ = (0, ..., 0, k-1, 0, ..., 0) \in S \]

2. $u \oplus_k v^{-1} = (0, ..., 0, k - 1, ..., k - 1) \in S$
3. $u \oplus_k v^{-1} = (0, ..., 0, 1, 0, ..., 0) \in S$
4. $u \oplus_k v^{-1} = (0, ..., 0, 1, ..., 1) \in S$.

Hence, $u \oplus_k v^{-1} \in S$.

Conversely, suppose that $u \oplus_k v^{-1} \in S$. So, $u \oplus_k v^{-1}$ is of the form $(0, ..., 0, 1, 0, ..., 0)$ or $(0, ..., 0, 1, 1, ..., 1)$ or $(0, ..., 0, k - 1, 0, ..., 0)$ or $(0, ..., 0, k - 1, 1, ..., k - 1)$.

Hence, for some $i$, one of the following holds:

1. $u = (u_n, ..., u_{i+1}, u_i \oplus_k (k - 1), u_{i-1}, ..., u_1)$
2. $v = (u_n, ..., u_{i+1}, u_i \oplus_k (k - 1), u_{i-1} \oplus_k (k - 1), ..., u_1 \oplus_k (k - 1))$
3. $v = (u_n, ..., u_{i+1}, u_i \oplus_k 1, u_{i-1}, ..., u_1)$
4. $v = (u_n, ..., u_{i+1}, u_i \oplus_k 1, u_{i-1} \oplus_k 1, ..., u_1 \oplus_k 1)$.

So $u$ and $v$ are adjacent in $AQ_{n,k}$.

As remarked earlier, (by definition) all Cayley graphs are vertex-symmetric and so we obtain an alternative proof of Corollary 5.3.2.

## 5.4 Connectivity

In this section, we examine the connectivity of $AQ_{n,k}$. By Menger's Theorem (see, for example, [21]), a graph $G = (V, E)$ has connectivity at least $c$ if, and only if,
5.4. Connectivity

Given any two distinct vertices of \( V \), there are \( c \) vertex-disjoint paths joining them. Having a high connectivity is a desirable property of any interconnection network as it provides fault-tolerance with regard to message routing, allows for hot-spots to be avoided, and allows large messages to be split up into smaller ones and routed in parallel along vertex-disjoint paths.

We show that \( \kappa(AQ_{n,k}) = 4n - 2 \), whenever \( n \geq 2 \) and \( k \geq 3 \). We begin by proving this result for \( AQ_{2,k} \) and then for the general case using a proof by induction (on \( n \)).

5.4.1 The base case of our induction

The base case of our forthcoming induction is provided by the following result.

**Lemma 5.4.1** The connectivity of \( AQ_{2,k} \) is 6; that is, \( \kappa(AQ_{2,k}) = 6 \).

**Proof:** We prove our result by constructing 6 disjoint paths joining any two distinct vertices of \( AQ_{2,k} \). By Lemma 5.3.1, w.l.o.g. we may suppose that our two given vertices of \( AQ_{2,k} \) are \( u = (0,0) \) and \( v = (i,j) \), where \( 0 \leq i \leq j \leq \lfloor \frac{k}{2} \rfloor \). For the case when \( k = 3 \), Lemma 5.3.1 tells us that we need only consider the cases when \( v \) is \((1,2)\) and \((2,2)\). The 6 disjoint paths between \((0,0)\) and \((1,2)\) are as follows:

1. \((0,0), (2, 2), (1, 2)\);
2. \((0,0), (2,0), (1,2)\);
3. \((0,0), (0,2), (1,2)\);
4. \((0,0), (0,1), (1,2)\);
5. \((0,0), (1,0), (1,2)\);
6. \((0,0), (1,1), (1,2)\).

The 6 disjoint paths between \((0,0)\) and \((2,2)\) are as follows:

1. \((0,0), (2,2)\);
2. \((0,0), (1,1), (2,2)\);
3. \((0,0), (0,2), (2,2)\);
4. \((0,0), (0,1), (1,2)\);
5. \((0,0), (1,0), (2,1), (2,2)\);
6. \((0,0), (1,1), (1,2)\).

For \( k > 3 \), we have 3 different cases to consider. Recall, \( 0 \leq i \leq j \leq \lfloor \frac{k}{2} \rfloor \leq k - 2 \).

**Case (i)** \( 0 < i < j \leq \lfloor \frac{k}{2} \rfloor \). Consider the following 6 paths:
5.4. Connectivity

Figure 5.2: The 6 disjoint paths when $0 < i < j$.

$\alpha_1$: $u, (k-1, 0), (k-2, 0), \ldots, (k-j+i, 0), (k-j+i-1, k-1), (k-j+i-2, k-2), \ldots, (i+1, j+1), v;$

$\alpha_2$: $u, (k-1, k-1), (k-2, k-2), \ldots, (j, j), (j-1, j), (j-2, j), \ldots, (i+1, j), v;$

$\alpha_3$: $u, (0, 1), (0, 2), \ldots, (0, j-i), (1, j-i+1), (2, j-i+2), \ldots, (i-1, j-1), v;$

$\alpha_4$: $u, (0, k-1), (0, k-2), \ldots, (0, j+1), (0, j), (1, j), (2, j), \ldots, (j-1, j), v;$

$\alpha_5$: $u, (1, 1), (2, 2), \ldots, (i, i), (i, i+1), (i, i+2), \ldots, (i, j-1), v;$

$\alpha_6$: $u, (1, 0), (2, 0), \ldots, (i, 0), (i, k-1), (i, k-2), \ldots, (i, j+1), v.$

These paths can be visualized in Fig. 5.2 and are clearly mutually disjoint (as there are no common nodes between them).

Case (ii) $0 < i = j \leq \left\lfloor \frac{k}{2} \right\rfloor$. Consider the following 6 paths:

$\alpha_1$: $u, (k-1, 0), (k-2, k-1), \ldots, (j, j+1), v;$
5.4. Connectivity

Figure 5.3: The 6 disjoint paths when $0 < i = j$.

$\alpha_2: u, (k - 1, k - 1), (k - 1, k - 2), \ldots, (j + 1, j + 1), v$;

$\alpha_3: u, (0, 1), (1, 2), (2, 3) \ldots, (j - 1, j), v$;

$\alpha_4: u, (0, k - 1), (k - 1, k - 2), (k - 2, k - 2), \ldots, (j + 1, j), v$;

$\alpha_5: u, (1, 1), (2, 2), \ldots, (j - 1, j - 1), v$;

$\alpha_6: u, (1, 0), (2, 1), (3, 2), \ldots, (j, j - 1), v$.

These paths can be visualized in Fig. 5.3 and are clearly mutually disjoint.

Case (iii) $i = 0$ and $1 \leq j \leq \lfloor \frac{3}{2} \rfloor$. Consider the following 6 paths:

$\alpha_1: u, (k - 1, 0), (k - 1, 1), \ldots, (k - 1, j - 1), v$;

$\alpha_2: u, (k - 1, k - 1), (k - 1, k - 2), \ldots, (k - 1, j), v$;

$\alpha_3: u, (0, 1), (0, 2), \ldots, (0, j - 1), v$;
These paths can be visualized in Fig. 5.4 and are clearly mutually disjoint. The result follows.

By examining each of the different constructions in the proof of Lemma 5.4.1, we see that the maximal length path joining \( u = (0, 0) \) and \( v = (i, j) \) is \( k \).

**Corollary 5.4.2** Given any two distinct vertices \( u \) and \( v \) of \( AQ_{2,k} \), there are 6 disjoint paths joining \( u \) and \( v \) so that the longest of these paths has length at most \( k \).

### 5.4.2 The induction step

We now prove our general connectivity result.

**Theorem 5.4.3** \( \kappa(AQ_{n,k}) = 4n - 2 \), whenever \( k \geq 3 \) and \( n \geq 2 \), and given any two distinct vertices of \( AQ_{n,k} \), there are \( 4n - 2 \) mutually disjoint paths joining these two vertices so that the length of the longest of these paths is at most \( \max\{(n - 1)k - (n - 2), k + 7\} \).

**Proof:** When \( n = 2 \) and \( k \geq 3 \), the result holds by Lemma 5.4.1. We proceed by induction on \( n \). Our induction hypothesis is that any two distinct vertices of \( AQ_{n-1,k} \) are joined by a set of \( 4n - 6 \) mutually disjoint paths (the base case of the induction is covered by Lemma 5.4.1).

We shall also calculate the length of a longest path as constructed according to this proof. Let \( d_n(w, w') \) be the maximal length of any path as constructed according to this proof joining any two vertices \( w \) and \( w' \) of \( AQ_{n,k} \), and let \( \delta_n = \max\{d_n(w, w') : w \text{ and } w' \text{ are distinct vertices of } AQ_{n,k}\} \). We shall obtain a recursive estimate of \( \delta_n \).

Fix \( k, n \geq 3 \). Given any two distinct vertices \( u \) and \( v \) of \( AQ_{n,k} \), we shall construct \( 4n - 2 \) disjoint paths joining them. By Lemma 5.3.1, w.l.o.g. we may assume that \( u = (0,0,\ldots,0) \) and \( v = (v_n, v_{n-1}, \ldots, v_1) \), with \( 0 \leq v_n \leq \lfloor \frac{k}{2} \rfloor \).
5.4. Connectivity

Figure 5.4: The 6 disjoint paths when $i = 0$.

Case 1: $v = (v_n, 0, 0, \ldots, 0)$, where $1 \leq v_n \leq \lfloor \frac{k}{2} \rfloor$; so, $v_n^0 = u$. The vertex $u$ has $4n - 6$ neighbours in $AQ_{n-1,k}^0$. For each of these neighbours $w$, apart from $(0, 1, 1, \ldots, 1)$ and $(0, k-1, k-1, \ldots, k-1)$, build the path from $w$ by traversing $(n, +1)$-edges until $AQ_{n-1,k}^{v_n}$ is reached, before moving to $v$. This accounts for $4n - 8$ mutually disjoint paths from $u$ to $v$. From the neighbour $(0, k-1, k-1, \ldots, k-1)$, build the path by traversing $(n, -1)$-edges until $AQ_{n-1,k}^{v_n}$ is reached, before moving to $v$. From the neighbour $(0, 1, 1, \ldots, 1)$, traverse $(n, -1)$-edges until $AQ_{n-1,k}^{v_n}$ is reached, before moving to $v$. This accounts for another 2 paths from $u$ to $v$ that are mutually disjoint and disjoint from all the other paths constructed above.

From the neighbour $(k-1, k-1, \ldots, k-1)$ of $u$, traverse $(n, -1)$-edges until $AQ_{n-1,k}^{v_n}$ is reached, before moving to $v$. From the neighbour $(1, 1, \ldots, 1)$ of $u$, traverse $(n, +1)$-edges until $AQ_{n-1,k}^{v_n}$ is reached, before moving to $v$. Finally, two additional paths are obtained by traversing $(n, +1)$-edges from $u$ until $v$ is reached, and by traversing $(n, -1)$-edges from $u$ until $v$ is reached. All paths constructed are mutually disjoint and can be visualized as in Fig. 5.5. Note that the length of the longest constructed path is $max\{v_n + 2, k - v_n + 1\}$; so, $d_n(u, v) \leq k$.

Having dealt with Case 1, let us henceforth assume that $v_n^0 \neq u$. We now define some paths which we shall use throughout the subsequent cases.

Our induction hypothesis is that there are $4n - 6$ disjoint paths joining any two distinct vertices of $AQ_{n-1,k}$. So, by our induction hypothesis, there is a set $\Pi$ of $4n - 6$ disjoint paths joining $u$ and $v_n^0$ in $AQ_{n-1,k}^0$ (by assumption $u$ and $v_n^0$ are
Let us denote 4 of these paths as follows:

- \( \pi_1 \) is the path passing through the neighbour \( u_{(\leq n-1,-1)} \) of \( u \);
- \( \pi_2 \) is the path passing through the neighbour \( u_{(\leq n-1,+1)} \) of \( u \);
- \( \pi_3 \) is the path passing through the neighbour \( v_{(\leq n-1,-1)}^{0} \) of \( v_{n}^{0} \);
- \( \pi_4 \) is the path passing through the neighbour \( v_{(\leq n-1,+1)}^{0} \) of \( v_{n}^{0} \).

Note that although \( \pi_1 \) and \( \pi_2 \) are always distinct, as are \( \pi_3 \) and \( \pi_4 \), it may be the case that either \( \pi_1 \) or \( \pi_2 \) is identical to either \( \pi_3 \) or \( \pi_4 \) (note also that any one of the above paths may consist of a solitary edge). We examine each of these circumstances separately. Moreover, there are two distinct situations: when \( v_n = 0 \); and when \( v_n \neq 0 \).
Note that every path $\pi$ in $\Pi$, from $u$ to $v|_n^0$, is such that there is a path $\pi^i$ in $AQ_{n-1,k}^{i}$, where $i \in \{1, 2, \ldots, k-1\}$, from $u|_n^i$ to $v|_n^i$, obtained by taking the isomorphic image of $\pi$ under the natural isomorphism (which takes $(0, a_{n-1}, a_{n-2}, \ldots, a_1)$ to $(i, a_{n-1}, a_{n-2}, \ldots, a_1)$; see Lemma 5.3.1). Throughout this proof, we extend this notation to arbitrary paths in $AQ_{n-1,k}^0$.

Consider the situation when $f = 0$ (and so $v^0_n = v$). For each path $\pi_j$, where $j \in \{1, 2, 3, 4\}$, that is not the path $u, v|_n^0$, truncate $\pi$ at the penultimate vertex (that is, the vertex of the path that is a neighbour of $v|_n^0$) and also remove the first edge: denote this truncated path by $\rho_j$ (note that a path might be truncated so that it consists of a solitary vertex). Do likewise with all isomorphic images of $\pi_1, \pi_2, \pi_3$ and $\pi_4$ (in $AQ_{n-1,k}^1, AQ_{n-1,k}^2$, and so on).

Suppose that $\rho_1 \neq \rho_3$. If neither $\rho_1$ nor $\rho_3$ is the path $u, v$ then we construct additional paths $u, \rho_1^{k-1}, v|_n^{k-1}, v$ and $u, u|_n^{k-1}, \rho_3^{k-1}, v$ through $AQ_{n-1,k}^{k-1}$. If $\rho_1 = u, v$ then we have that $v = (0, k-1, k-1, \ldots, k-1)$. In this case, we construct additional paths $u, u|_n^{k-1}, \rho_3^{k-1}, v$ and $u, v|_n^{k-1}, v$ through $AQ_{n-1,k}^{k-1}$. If $\rho_3 = u, v$ then we have that $u = (0, v_{n-1} - 1, v_{n-2} - 1, \ldots, v_1 - 1)$, with $v_{n-1} = v_{n-2} = \ldots = v_1 = 1$. In this case, we construct additional paths $u, u|_n^{k-1}, v$ and $u, \rho_1^{k-1}, v|_n^{k-1}, v$ through $AQ_{n-1,k}^{k-1}$.

Suppose that $\rho_1 = \rho_3$. We have that $\rho_1 \neq \rho_2$. In this case, we construct additional paths $u, \rho_1^{k-1}, v$ and $u, u|_n^{k-1}, \rho_2^{k-1}, v|_n^{k-1}, v$ through $AQ_{n-1,k}^{k-1}$.

We proceed in an analogous fashion by considering $\rho_2$ and $\rho_4$ in the same way, and constructing disjoint paths from $u$ to $v$ through $AQ_{n-1,k}^1$. Consequently, we obtain $4n - 2$ disjoint paths from $u$ to $v$ in $AQ_{n,k}$. From the above construction, we clearly have that $d_n(u, v) = d_{n-1}((0, 0, \ldots, 0), (v_{n-1}, v_{n-2}, \ldots, v_1)) + 2 \leq \delta_{n-1} + 2$.

Henceforth, we shall assume that $v_n \neq 0$.

**Case 2:** $u \neq v|_n^0$, $u$ is not adjacent to $v|_n^0$, and $u$ and $v|_n^0$ do not have a neighbour of $AQ_{n-1,k}^0$ in common.

In particular, $u, v$ is not a path in $\Pi$.

**Sub-case 2.1:** $\rho_1 \neq \rho_4$ and $\rho_2 \neq \rho_3$.

We begin by building 6 specific paths:

$$\alpha_1: u, \rho_1^{k-1}, v|_n^{k-1}, v|_n^{k-2}, \ldots, v|_n^{v_{n+1}}, v;$$
5.4. Connectivity

\[ \alpha_2: \ u, u^{n-1}_n, p_4^{n-1}_n, v^{(\leq n+1)}_n, u^{(\leq n+1)}_n, v^{(\leq n+1)}_n, \ldots, u^{(\leq n+1)}_n, v; \]

\[ \alpha_3: \ u, u^{1}_n, u^{2}_n, \ldots, u^{v_n}_n, p_3^{v_n}, v; \]

\[ \alpha_4: \ u, u^{(\leq n+1)}_n, u^{(\leq n+1)}_n, v^{(\leq n+1)}_n, u^{v_n-1}_n, \ldots, u^{(\leq n+1)}_n, v; \]

\[ \alpha_5: \ u, p_2, p_2^{0}, u^{1}_n, \ldots, v^{1}_n, v; \]

\[ \alpha_6: \ u, p_3, v^{(\leq n-1)}_n, v^{(\leq n-1)}_n, \ldots, v^{(\leq n-1)}_n, v. \]

These paths can be visualized as in Fig. 5.6, and can easily be seen to be mutually disjoint.

There are \( 4n - 8 \) paths in \( \Pi \) apart from \( \pi_2 \) and \( \pi_3 \); let \( \pi \) be any one of them. We truncate \( \pi \) at the penultimate vertex, and then extend this path along \((n, +1)\)-edges until we reach \( AQ^{v_n}_{n-1,k} \). Finally, we extend the path by an edge to \( v \). Again, it is easy to see that the resulting set of \( 4n - 2 \) paths are mutually disjoint. Furthermore, we have that \( d_n(u, v) = d_{n-1}((0, 0, \ldots, 0), (v_{n-1}, v_{n-2}, \ldots, v_1)) + \max\{k - v_n - 1, v_n\} \leq \delta_{n-1} + k - 2. \)

**Sub-case 2.2:** \( \rho_1 = p_4 \) and \( \rho_2 \neq p_3. \)

Note that, by definition, \( \rho_1, \rho_2 \) and \( \rho_3 \) are distinct. Referring to Sub-case 2.1 (and Fig. 5.6), if we can amend paths \( \alpha_1 \) and \( \alpha_2 \) so that they remain disjoint and also disjoint from all of the other \( 4n - 4 \) paths then we are done. Replace \( \alpha_1 \) and \( \alpha_2 \) with the paths \( \alpha'_1 \) and \( \alpha'_2 \) defined as:

\[ \alpha'_1: \ u, p_1^{k-1}, v^{(\leq n+1)}_n, v^{(\leq n+1)}_n, v^{(\leq n+1)}_n, v; \]

\[ \alpha'_2: \ u, u^{(\leq n+1)}_n, \rho_2^{k-1}, v^{(\leq n+1)}_n, v^{(\leq n+1)}_n, v; \]

Again, it is easy to see that the resulting set of \( 4n - 2 \) paths are mutually disjoint. The amendments made can be visualized as in Fig. 5.7. Furthermore, we have that \( d_n(u, v) = d_{n-1}((0, 0, \ldots, 0), (v_{n-1}, v_{n-2}, \ldots, v_1)) + \max\{k - v_n, v_n\} \leq \delta_{n-1} + k - 1. \)

**Sub-case 2.3:** \( \rho_1 \neq p_4 \) and \( \rho_2 = p_3. \)

Note that, by definition, \( \rho_1, \rho_2 \) and \( \rho_4 \) are distinct. Referring to Sub-case 2.1 (and Fig. 5.6), if we can amend paths \( \alpha_3, \alpha_4, \alpha_5 \) and \( \alpha_6 \) so that they remain disjoint and also disjoint from all of the other \( 4n - 6 \) paths then we are done. Replace \( \alpha_3, \alpha_4, \alpha_5 \) and \( \alpha_6 \) with the paths \( \alpha'_3, \alpha'_4, \alpha'_5 \) and \( \alpha'_6 \) defined as:
5.4. Connectivity

Figure 5.6: The 6 disjoint paths in Sub-case 2.1.

Figure 5.7: The amendments in Sub-case 2.2.
\[\alpha_3: u, u_{n-1}^{\leq 1}, u_{n-1}^{\leq 2}, \ldots, u_{n-1}^{\leq v_n}, \rho_{n}^{v_n}, v;\]
\[\alpha_4: u, u_{n-1}^{\leq u}, u_{n-1}^{\leq u}, \ldots, u_{n-1}^{\leq v_n}, \rho_{n}^{v_n}, v;\]
\[\alpha_5: u, \rho_2, v_{n-1}, v_{n-2}, \ldots, v_{n-1}, v;\]
\[\alpha_6: u, \rho_1, v_{n}^{0}, v_{n}^{1}, \ldots, v_{n}^{v_n-1}, v.\]

Again, it is easy to see that the resulting set of \(4n - 2\) paths are mutually disjoint. The amendments made can be visualized as in Fig. 5.8. Furthermore, we have that
\[d_n(u, v) = d_{n-1}((0, 0, \ldots, 0), (v_{n-1}, v_{n-2}, \ldots, v_1)) + \max\{k - v_n - 1, v_n\} \leq \delta - n + k - 2.\]

Sub-case 2.4: \(\rho_1 = \rho_4\) and \(\rho_2 = \rho_3.\)

By making the amendments in Sub-cases 2.2 and 2.3, we obtain a set of \(4n - 2\) mutually disjoint paths. Furthermore, we have that
\[d_n(u, v) = d_{n-1}((0, 0, \ldots, 0), (v_{n-1}, v_{n-2}, \ldots, v_1)) + \max\{k - v_n, v_n\} \leq \delta_n - 1 + k - 1.\]

Case 3: \(u \neq v_n^0\) and \(u\) and \(v_n^0\) are not adjacent, but \(u\) and \(v_n^0\) have a neighbour of \(AQ_{n-1, k}\) in common.

All the constructions in Sub-cases 2.1, 2.2, 2.3 and 2.4 work here unless \((v_{n-1} - 1, v_{n-2} - 1, \ldots, v_1 - 1) = (1, 1, \ldots, 1), i.e., unless \(v = (v_n, 2, 2, \ldots, 2).\) Thus, this is the only situation to deal with (note that \(k \geq 4,\) as otherwise \(u\) and \(v_n^0\) would be adjacent).

One of the paths in the set \(\Pi\) is the path \((0, 1, 1, \ldots, 1), v,\) and let \(\pi\) be the path passing through \((0, 3, 3, \ldots, 3).\) Truncate \(\pi\) at the penultimate vertex \((0, 3, 3, \ldots, 3)\) and also remove the first edge: denote this truncated path by \(\rho\) (note that the path \(\rho\) might consist of the solitary vertex \((0, 3, 3, \ldots, 3)\)). Define the paths \(\rho^i,\) for \(i \in \{1, 2, \ldots, k - 1\},\) as we did earlier.

Sub-case 3.1: \(v_n > 1.\)

We begin by building 6 specific paths:

\[\alpha_1: u, \rho^{k-1}, v_{n-1}^{\leq k-2}, v_{n-1}^{\leq k-3}, \ldots, v_{n-1}^{\leq v_n}, v;\]
\[\alpha_2: u, u_{n-1}^{k-1}, v_{n-1}^{\leq k-1}, v_{n-1}^{k-2}, \ldots, v_{n-1}^{v_n+1}, v;\]
\[\alpha_3: u, u_{n-1}^{1}, u_{n-1}^{2}, \ldots, u_{n-1}^{v_n}, v_{n-1}^{\leq v_n}, v;\]
Figure 5.8: The amendments in Sub-case 2.3.
5.4. Connectivity

Figure 5.9: The paths in Sub-case 3.1.

\[ \alpha_4: u, v, v(\leq n, -1)_{n1}, v(\leq n, -1)_{n2}, \ldots, v(\leq n, -1)_{nk-1}, v; \]
\[ \alpha_5: u, \rho, v(\leq n, +1)_{n1}, v(\leq n, +1)_{n2}, \ldots, v(\leq n, +1)_{nk-1}, v; \]
\[ \alpha_6: u, v(\leq n, -1)_{n0}, v_{n1}, v_{n2}, \ldots, v_{nk-1}, v. \]

These paths can be visualized as in Fig. 5.9, and can easily be seen to be disjoint.

There are \( 4n - 8 \) paths in \( \Pi \) apart from \( \pi \) and \( (0, 1, 1, \ldots, 1), v \); let \( \pi' \) be any one of them. We truncate \( \pi' \) at the penultimate vertex, and then extend this path along \( (n, +1) \)-edges until we reach \( AQ_{n-1,k}^k \). Finally, we extend the path by an edge to \( v \).

Again, it is easy to see that the resulting set of \( 4n - 2 \) paths are mutually disjoint.

Furthermore, we have that \( d_n(u, v) = d_{n-1}((0, 0, \ldots, 0), (2, 2, \ldots, 2)) + \max\{k - v_n - 2, v_n\} \leq \delta_{n-1} + \max\{k - 4, \left[ k \frac{3}{2} \right]\}. \)
Sub-case 3.2: \( v_n = 1 \).

We begin by building 6 specific paths:

- \( \alpha_1: u, \rho^{k-1}, v_{(\leq n, +1)}|^{k-2}, v_{(\leq n, +1)}|^{k-3}, \ldots, v_{(\leq n, +1)} |^{k} \); \\
- \( \alpha_2: u, v_{(\leq n, +1)}|^{k-1}, v_{(\leq n, +1)} |^{k-2}, v_{(\leq n, +1)} |^{k-3}, \ldots, v_{(\leq n, +1)} |^{k} \); \\
- \( \alpha_3: u, v_{(\leq n, +1)} |^{k} \); \\
- \( \alpha_4: u, v_{(\leq n, +1)} |^{k} \); \\
- \( \alpha_5: u, v_{(\leq n, +1)} |^{k} \); \\
- \( \alpha_6: u, v_{(\leq n, +1)} |^{k} \).

These paths can be visualized as in Fig. 5.10, and can easily be seen to be mutually disjoint. There are \( 4n - 8 \) paths in \( \Pi \) apart from \( \pi \) and \( u, (0, 1, 1, \ldots, 1), v \); let \( \pi' \) be any one of them. We truncate \( \pi' \) at the penultimate vertex, and then extend this path along an \( (n, +1) \)-edge and then an edge to \( v \). Again, it is easy to see that the resulting set of \( 4n - 2 \) paths are mutually disjoint. Furthermore, we have that \( d_{\pi}(u, v) = \max \{d_{\pi-1}((0, 0, 0, \ldots, 0), (2, 2, 2, \ldots, 2)) + k - 3, k + 1 \} \leq \delta_{n-1} + k - 3. \)

Case 4: \( u \) and \( v_{(\leq n, +1)} \) are adjacent.

Sub-case 4.1: \( v_{(\leq n, +1)} \not\subseteq \{(0, k - 1, k - 1, \ldots, k - 1), (0, k - 1, \ldots, 1), (0, 2, 2, \ldots, 2)\} \).

Note that as \((0, k - 1, k - 1, \ldots, k - 1) \neq v_{(\leq n, +1)} \neq (0, 1, 1, \ldots, 1)\), none of the vertices \((0, k - 1, k - 1, \ldots, k - 1), (0, 1, 1, \ldots, 1), (0, v_{n-1} - 1, v_{n-2} - 1, \ldots, v_{1} - 1)\) and \((0, v_{n-1} + 1, v_{n-2} + 1, \ldots, v_{1} + 1)\) is identical to either \( u \) or \( v_{(\leq n, +1)} \). Note also that as \( u \) and \( v_{(\leq n, +1)} \) are adjacent, so are \((i, 1, 1, \ldots, 1)\) and \((i, v_{n-1} + 1, v_{n-2} + 1, \ldots, v_{1} + 1)\) and also \((i, k - 1, k - 1, \ldots, k - 1)\) and \((i, v_{n-1} - 1, v_{n-2} - 1, \ldots, v_{1} - 1)\), for \( i \in \{1, 2, \ldots, k - 1\} \).

One of the paths in \( \Pi \) is the edge \((u, v_{(\leq n, +1)})\). For each path in \( \Pi \), apart from the edge \((u, v_{(\leq n, +1)})\) and the path passing through \((0, v_{n-1} - 1, v_{n-2} - 1, \ldots, v_{1} - 1)\), truncate this path at the penultimate vertex and extend it using \((n, +1)\)-edges until \( AQ_{n-1,k}^{v_{n-1}} \) is reached before extending it further by an edge to \( v \). As regards the path in \( \Pi \) passing through \((0, v_{n-1} - 1, v_{n-2} - 1, \ldots, v_{1} - 1)\), truncate this path at \((0, v_{n-1} - 1, v_{n-2} - 1, \ldots, v_{1} - 1)\) and extend it using \((n, +1)\)-edges until \( AQ_{n-1,k}^{v_{n-1}-1} \) is
5.4. Connectivity

reached before extending it further by an edge to $v$. Also, extend the edge $(u, v|_0^0)$ using $(n, +1)$-edges to $v$. These $4n - 6$ paths from $u$ to $v$ can be visualized as in Fig. 5.11.

Form the following paths:

$\alpha_1$: $u, u|_1^{n}, u|_1^{n+1}, v|_1^{n+1}, v|_1^{n+1}$;

$\alpha_2$: $u, v|_n^{1}, u|_n^{2}, \ldots, u|_n^{v_n}, v$;

$\alpha_3$: $u, v|_n^{k-n}, v|_n^{k-1}, v|_n^{k-2}, \ldots, v|_n^{v_n}, v$;

$\alpha_4$: $u, v|_n^{0}, u|_n^{v_n}, u|_n^{v_n}, v|_n^{v_n}, v$.

All paths can be visualized in Fig. 5.11. It is easy to see that as $(0, 1, 1, \ldots, 1) \neq (0, v_{n-1} - 1, v_{n-2} - 1, \ldots, v_1 - 1)$, i.e., $v|_n^0 \neq (0, 2, 2, \ldots, 2)$, the $4n - 6$ paths, constructed above, and the paths $\alpha_1, \alpha_2, \alpha_3$ and $\alpha_4$ are all mutually disjoint. Furthermore, we have that $d_n(u, v) = max\{d_{n-1}((0, 0, \ldots, 0), (v_{n-1}, v_{n-2}, \ldots, v_1)) + v_n, k - v_n + 2, v_n + 3\} \leq \delta_{n-1} + \lfloor \frac{k}{2} \rfloor$.

Sub-case 4.2: $v|_n^0 = (0, 1, 1, \ldots, 1)$. 

Figure 5.10: The paths in Sub-case 3.2.
Figure 5.11: The paths in Sub-case 4.1.
One of the paths in \( \Pi \) is the edge \((u, v_n^0)\). For each path in \( \Pi \), apart from the edge \((u, v_n^0)\), truncate this path at the penultimate vertex and extend it using \((n, +1)\)-edges until \(AQ^m_{n-1,k}\) is reached before extending it further by an edge to \(v\). Extend the edge \((u, v_n^0)\) using \((n, -1)\)-edges to \(v\).

Let the path \(\rho\) in \(AQ^{k-1}_{n-1,k}\) be defined as \((k - 1, k - 1, k - 1, \ldots, k - 1), (k - 1, 0, k - 1, \ldots, k - 1), (k - 1, 1, k - 1, \ldots, k - 1), (k - 2, k - 1, \ldots, k - 1), (k - 1, 2, 0, \ldots, 0), (k - 1, 2, 1, \ldots, 1), (k - 1, 2, 2, \ldots, 2)\) (unless \((k - 1, k - 1, k - 1, \ldots, k - 1) = (k - 1, 2, 2, \ldots, 2)\) when \(\rho\) is just a solitary vertex). Note that \(\rho\) avoids \((k - 1, 0, 0, \ldots, 0)\) and \((k - 1, 1, 1, \ldots, 1)\). Define the paths:

\[
\alpha_1: u, \rho, v_{(\leq n, +1)}^{k-2}, v_{(\leq n, +1)}^{k-3}, \ldots, v_{(\leq n, +1)}, v;
\]
\[
\alpha_2: u, v_n^{k-1}, u_n^{k-2}, \ldots, u_n, v;
\]
\[
\alpha_3: u, u_n^1, u_n^2, \ldots, u_n^{v_n^{-1}}, v;
\]
\[
\alpha_4: u, v_n^0, v_n^1, \ldots, v_n^{v_n^{-1}}, v.
\]

Our collection of \(4n - 2\) paths from \(u\) to \(v\) can be visualized as in Fig. 5.12, and from the above construction, they are clearly mutually disjoint. Furthermore, we have that 

\[d_n(u, v) = \max\{d_{n-1}((0, 0, 0, \ldots, 0), (1, 1, \ldots, 1)) + v_n, k - v_n + 6\} \leq \max\{\delta_{n-1} + \left\lfloor \frac{k}{2} \right\rfloor, k + 5\}.
\]

Sub-case 4.3: \(v_n^0 = (0, k - 1, k - 1, \ldots, k - 1)\).

One of the paths in \(\Pi\) is the edge \((u, v_n^0)\). For each path in \(\Pi\), apart from the edge \((u, v_n^0)\) and the paths passing through \((0, 1, 1 \ldots, 1)\) and \((0, k - 2, k - 2, \ldots, k - 2)\), truncate this path at the penultimate vertex and extend it using \((n, +1)\)-edges until \(AQ^{m}_{n-1,k}\) is reached before extending it further by an edge to \(v\). Extend the edge \((u, v_n^0)\) using \((n, +1)\)-edges to \(v\), and extend the truncated path through \((0, k - 2, k - 2, \ldots, k - 2)\) using \((n, +1)\)-edges to \((v_n - 1, k - 2, k - 2, \ldots, k - 2)\) and then to \(v\). This accounts for \(4n - 7\) mutually disjoint paths.

Let the path \(\rho\) in \(AQ^{m+1}_{n-1,k}\) be defined as \((v_n + 1, k - 2, k - 2, \ldots, k - 2), (v_n + 1, k - 1, k - 2, \ldots, k - 2), (v_n + 1, 0, k - 2, \ldots, k - 2), (v_n + 1, 1, k - 2, \ldots, k - 2), (v_n + 1, 1, k - 1, \ldots, k - 1), (v_n + 1, 1, 0, \ldots, 0), (v_n + 1, 1, 1, \ldots, 1)\) (unless \((v_n + 1, k - 2, k - 2, \ldots, k - 2)\) when \(\rho\) is just a solitary vertex). Note that \(\rho\) avoids \((0, 0, \ldots, 0)\) and \((1, 1, \ldots, 1)\). Define the paths:

\[
\alpha_1: u, \rho, v_{(\leq n, +1)}^{k-2}, v_{(\leq n, +1)}^{k-3}, \ldots, v_{(\leq n, +1)}, v;
\]
\[
\alpha_2: u, v_n^{k-1}, u_n^{k-2}, \ldots, u_n, v;
\]
\[
\alpha_3: u, u_n^1, u_n^2, \ldots, u_n^{v_n^{-1}}, v;
\]
\[
\alpha_4: u, v_n^0, v_n^1, \ldots, v_n^{v_n^{-1}}, v.
\]
Figure 5.12: The paths in Sub-case 4.2.
Figure 5.13: The paths in Sub-case 4.3.

\[ 2, \ldots, k - 2 = (v_n + 1, 1, 1, \ldots, 1) \text{ when } \rho \text{ is just a solitary vertex}. \] Note that \( \rho \) avoids \((v_n + 1, 0, 0, \ldots, 0)\) and \((v_n + 1, k - 1, k - 1, \ldots, k - 1)\). Define the paths:

\[ \alpha_1: u, u|n^{k-1}, u|n^{k-2}, \ldots, u|n^{v_n+1}, v; \]

\[ \alpha_2: u, v|n^{k-1}, v|n^{k-2}, \ldots, v|n^{v_n+1}, v; \]

\[ \alpha_3: u, \{u(\leq n+1), u(\leq n+1)\}_{n}^{2}, u(\leq n+1)_{n}, \ldots, u(\leq n+1)_{v_n}, v; \]

\[ \alpha_4: u, u|n^{1}, u|n^{2}, \ldots, u|n^{v_n}, v; \]

\[ \alpha_5: u, u(\leq n+1)_{n}^{0}, u(\leq n+1)_{n}^{k-1}, u(\leq n+1)_{n}^{k-2}, \ldots, u(\leq n+1)_{n}^{v_n+2}, \rho, v(\leq n-1)_{n}^{v_n}, v. \]

Our collection of \( 4n - 2 \) paths from \( u \) to \( v \) can be visualized as in Fig. 5.13, and they are clearly mutually disjoint. Furthermore, we have that \( d_n(u, v) = \)
Sub-case 4.4: \( v^0_n = (0, 2, 2, \ldots, 2) \).

As \( u \) and \( v^0_n \) are adjacent, we must have that \( k = 3 \) and that \( v = (1, 2, 2, \ldots, 2) \).

By Lemma 5.3.1, there exists an automorphism of \( AQ_{n,k} \) mapping \((1, 2, 2, \ldots, 2)\) to \((2, 1, 1, \ldots, 1)\) and fixing \( u \). Thus, this sub-case reduces to Sub-case 4.2.

As regards the length of the longest path constructed, we have that \( \delta_n \leq \max\{\delta_{n-1} + k - 1, k + 7\} \) and \( \delta_2 = k \). Thus, \( \delta_n \leq (n - 1)k - (n - 2) \), unless: 
\( n = 3 \) and \( k = 3, 4, 5, 6, 7 \); \( n = 4 \) and \( k = 3, 4 \); or \( n = 5 \) and \( k = 3 \), when \( \delta_n \leq k + 7 \). 

The result follows by induction. \( \square \)

5.5 The diameter

Obviously, the smaller the diameter of an interconnection network, the lower the communication latency (be this under store-and-forward or wormhole routing). In this section, we obtain the diameter of \( AQ_{2,k} \) and an upper bound on the diameter of \( AQ_{n,k} \) when \( n \geq 3 \).

We begin with some immediate observations as regards the order of edges in paths in \( AQ_{n,k} \). Consider some path \( \rho \) from some vertex \( u \) of \( AQ_{n,k} \) to some vertex \( v \) of \( AQ_{n,k} \) within which there is an \( \lambda \)-edge, where \( \lambda \in \{(i,+1), (i,-1), (\leq i,+1), (\leq i,-1)\} \), for some \( i \), as the \( a \)-th edge of the path, and a \( \mu \)-edge, where \( \mu \in \{(j,+1), (j,-1), (\leq j,+1), (\leq j,-1)\} \), for some \( j \), as the \( b \)-th edge of the path, where \( a \neq b \). The path obtained from \( \rho \) by traversing a \( \mu \)-edge as the \( a \)-th edge of the path and a \( \lambda \)-edge as the \( b \)-th edge of the path, and leaving the labels of all other edges as they were, is still a path from \( u \) to \( v \). Also, if \( \rho \) is a shortest path between \( u \) and \( v \) and there is a \((i,+1)\)-edge (resp. \((i,-1)\)-edge, \((\leq i,+1)\)-edge, \((\leq i,-1)\)-edge) in \( \rho \), for some particular \( i \), then there is no \((i,-1)\)-edge (resp. \((i,+1)\)-edge, \((\leq i,-1)\)-edge, \((\leq i,+1)\)-edge) in \( \rho \). We use these observations throughout the proof of the following result.

Proposition 5.5.1 The diameter of \( AQ_{2,k} \) is \( \lceil \frac{k}{3} \rceil + \lceil \frac{k-1}{3} \rceil \), and for \( n \geq 3 \) the diameter of \( AQ_{n,k} \) is at most \( \frac{k}{4}(n + 1) \), if \( k \) is even, and at most \( \frac{k}{4}(n + 1) + \frac{n}{4} \), if \( k \) is
5.5. The diameter

odd.

Proof: By Corollary 5.3.2, we may restrict our attention to the lengths of paths from an arbitrary vertex of $AQ_{n,k}$ to the vertex $u = (0,0,\ldots,0)$ of $AQ_{n,k}$ when determining the diameter of $AQ_{n,k}$.

Let $v = (v_2,v_1)$ be a vertex of $AQ_{2,k}$.

Case (i): $k \equiv 0 \pmod{3}$.

Sub-case (a): $v_1, v_2 \notin \{\frac{k}{3} + 1, \frac{k}{3} + 2, \ldots, \frac{2k}{3} - 1\}$.

By traversing edges with labels from $\{(i, 1), (i, -1) : i = 1, 2, \ldots, n\}$, we can obtain a path of length at most $\frac{2k}{3}$ from $v$ to $u$.

Sub-case (b): exactly one of $v_1$ and $v_2$ is in $\{\frac{k}{3} + 1, \frac{k}{3} + 2, \ldots, \frac{2k}{3} - 1\}$.

Suppose that $v_1 \in \{\frac{k}{3} + 1, \frac{k}{3} + 2, \ldots, \frac{2k}{3} - 1\}$. By traversing $(1, 1)$-edges or $(1, -1)$-edges, we can move from $v$ to $(v_2, v_1)$, and by traversing $(2, 1)$-edges or $(2, -1)$-edges we can then move to $u$. This yields a path of length at most $\frac{2k}{3} - 1$ from $v$ to $u$.

If $v_2 \in \{\frac{k}{3} + 1, \frac{k}{3} + 2, \ldots, \frac{2k}{3} - 1\}$ then we proceed similarly except that we first traverse $(2, 1)$-edges or $(2, -1)$-edges to get to $(v_1, v_2)$, before traversing $(1, 1)$-edges or $(1, -1)$-edges to get to $u$.

Sub-case (c): $v_1, v_2 \in \{\frac{k}{3} + 1, \frac{k}{3} + 2, \ldots, \frac{2k}{3} - 1\}$.

Proceeding similarly to as in Sub-case (b) results in a path from $v$ to $u$ of length at most $\frac{2k}{3} - 1$.

In consequence, when $k \equiv 0 \pmod{3}$ there is a path from $v$ to $u$ of length at most $\frac{2k}{3} = \lceil \frac{k}{3} \rceil + \lceil \frac{k-1}{3} \rceil$.

Case (ii): $k \equiv 1 \pmod{3}$.

We proceed similarly to as in Case (i) except that we consider the values of $v_1$ and $v_2$ as to whether they lie in $\{\lceil \frac{k}{3} \rceil + 1, \lceil \frac{k}{3} \rceil + 2, \ldots, \lceil \frac{k}{3} \rceil + \lceil \frac{k-1}{3} \rceil - 1\}$. We thus obtain a path from $v$ to $u$ of length at most $\lceil \frac{k}{3} \rceil + \lceil \frac{k-1}{3} \rceil - 1 = \lceil \frac{k}{3} \rceil + \lceil \frac{k-1}{3} \rceil$.

Case (iii): $k \equiv 2 \pmod{3}$.

We proceed similarly to as in Case (i) except that we consider the values of $v_1$ and $v_2$ as to whether they lie in $\{\lceil \frac{k}{3} \rceil + 1, \lceil \frac{k}{3} \rceil + 2, \ldots, 2\lceil \frac{k}{3} \rceil - 1\}$. We thus obtain a path
from $v$ to $u$ of length at most $\left\lfloor \frac{k}{3} \right\rfloor + \left\lfloor \frac{k-1}{3} \right\rfloor$. In consequence, when $k \equiv 2$ there is a path from $v$ to $u$ of length at most $\left\lfloor \frac{k}{3} \right\rfloor + \left\lfloor \frac{k-1}{3} \right\rfloor = \left\lfloor \frac{k}{3} \right\rfloor + \left\lfloor \frac{k-4}{3} \right\rfloor$.

Whilst $\left\lfloor \frac{k}{3} \right\rfloor + \left\lfloor \frac{k-1}{3} \right\rfloor$ is an upper bound on the diameter of $AQ_{2,k}$, it is also a lower bound as we now show. Suppose that $k \equiv 0 \pmod{3}$ and the length of a shortest path $\rho$ from $(\frac{k}{3}, \frac{2k}{3})$ to $(0,0)$ is less than $\left\lfloor \frac{k}{3} \right\rfloor + \left\lfloor \frac{k-1}{3} \right\rfloor = \frac{2k}{3}$. If the edges of $\rho$ are all $(i,+1)$-edges or $(i,-1)$-edges then we immediately obtain a contradiction. Thus, there must be some $(\leq 2, +1)$-edges or $(\leq 2, -1)$-edges in $\rho$. By symmetry, we may suppose that there are $(\leq 2, -1)$-edges (and so, as $\rho$ is a shortest path, there must be no $(\leq 2, +1)$-edges in $\rho$). Moreover, based on our observation, we may clearly assume that all these $(\leq 2, -1)$-edges appear as a prefix of $\rho$.

Suppose that there are at most $\frac{2k}{3} - \left\lfloor \frac{k}{2} \right\rfloor$ $(\leq 2, -1)$-edges in $\rho$ and that traversing these $(\leq 2, -1)$-edges takes us to $(v_2', v_1')$. For an arbitrary vertex $(v_2, v_1)$ of $AQ_{2,k}$, define $wt(v_2, v_1) = \min\{v_2, k - v_2\} + \min\{v_1, k - v_1\}$, i.e., the distance of $(v_2, v_1)$ from $(0,0)$ in the $k$-ary 2-cube $Q_k$. As $wt(\frac{k}{3}, \frac{2k}{3}) = wt(v_2', v_1') = \frac{2k}{3}$, this yields a contradiction (as any path from $(v_2', v_1')$ to $(0,0)$ traversing only edges with labels from $\{(1,+1), (1,-1), (2,+1), (2,-1)\}$ has length at least $wt(v_2', v_1')$). Thus there must be between $\frac{2k}{3} - \left\lfloor \frac{k}{2} \right\rfloor + 1$ and $\frac{k}{3} (\leq 2, -1)$-edges in $\rho$ (clearly there cannot exist more than $\frac{k}{3}$ such edges as otherwise we could obtain a shorter path than $\rho$).

Suppose that there exist $m + \frac{2k}{3} - \left\lfloor \frac{k}{2} \right\rfloor$ $(\leq 2, -1)$-edges in $\rho$, where $1 \leq m \leq \left\lfloor \frac{k}{2} \right\rfloor - \frac{k}{3}$, and that traversing these edges takes us to the vertex $(v_2', v_1')$. Then $wt(v_2', v_1') = \frac{2k}{3} - 2(m-1) - 1$. Any path from $(v_2', v_1')$ to $(0,0)$ not using $(\leq 2, +1)$-edges nor $(\leq 2, -1)$-edges has length at least $\frac{2k}{3} - 2(m-1) - 1$. Thus, the length of $\rho$ is at least $\left(\frac{2k}{3} - 2(m-1) - 1\right) + (m + \frac{2k}{3} - \left\lfloor \frac{k}{2} \right\rfloor) = \frac{4k}{3} - m + 1 - \left\lfloor \frac{k}{2} \right\rfloor \geq \frac{4k}{3} - \left(\frac{k}{2} - \frac{k}{2}\right) + 1 - \left\lfloor \frac{k}{2} \right\rfloor = \frac{4k}{3} - 2\left\lfloor \frac{k}{2} \right\rfloor + 1 = \frac{2k}{3}$, which yields a contradiction.

Arguing in an analogous fashion with the vertex $(\left\lfloor \frac{k}{3} \right\rfloor, \left\lfloor \frac{k}{3} \right\rfloor + \left\lfloor \frac{k}{3} \right\rfloor)$ of $AQ_{2,k}$, when $k \equiv 1 \pmod{3}$, and with the vertex $(\left\lfloor \frac{k}{3} \right\rfloor, 2\left\lfloor \frac{k}{3} \right\rfloor)$ of $AQ_{2,k}$, when $k \equiv 2 \pmod{3}$, yields that the diameter of $AQ_{2,k}$ is $\left\lfloor \frac{k}{3} \right\rfloor + \left\lfloor \frac{k-1}{3} \right\rfloor$ irrespective of the value of $k \pmod{3}$.

Let $n \geq 3$ and $v = (v_n, v_{n-1}, \ldots, v_1)$ be a vertex of $AQ_{n,k}$.

**Case (i):** $k$ is even.

Define $\Sigma_{i=1}^n \frac{k}{2} - v_i = \alpha$. Traversing $\frac{k}{2}$ $(\leq n, -1)$-edges from $v$ leads to a vertex $v' = (v'_n, v'_{n-1}, \ldots, v'_1)$ such that $\Sigma_{i=1}^n \min\{v'_i, k - v'_i\} = \alpha$, and so by traversing
(i,+1)-edges and (i,-1)-edges, for various i, as appropriate, we obtain a path of length \( \frac{k}{2} + \alpha \) from \( v \) to \( u \). Alternatively, we could simply start from \( v \) and traverse (i,+1)-edges and (i,-1)-edges, as appropriate, to obtain a path of length \( \frac{nk}{2} - \alpha \) from \( v \) to \( u \).

Suppose that \( \frac{k}{2} + \alpha \leq \frac{nk}{2} - \alpha \), i.e., \( 2\alpha \leq \frac{k}{2}(n - 1) \). So, there is a path of length at most \( \frac{k}{2} + \frac{k}{2}(n - 1) = \frac{k}{2}(n + 1) \) from \( v \) to \( u \). If \( 2\alpha > \frac{k}{2}(n - 1) \) then there is a path of length less than \( \frac{nk}{2} - \frac{k}{2}(n - 1) = \frac{k}{2}(n + 1) \) from \( v \) to \( u \). Thus, when \( k \) is even there is a path of length at most \( \frac{k}{2}(n + 1) \) from \( v \) to \( u \).

**Case (ii):** \( k \) is odd.

We proceed similarly to as in Case (i) but the numerics are slightly messier. Define \( \Sigma_{i=1}^{n}[\lceil \frac{k}{2} \rceil] - v_i = \alpha \). Similarly to as in Case (i), we obtain a path from \( v \) to \( u \) of length at most \( \lceil \frac{k}{2} \rceil + \alpha \) and also one of length at most \( n\lceil \frac{k}{2} \rceil - \alpha \).

Suppose that \( \lceil \frac{k}{2} \rceil + \alpha \leq n\lceil \frac{k}{2} \rceil - \alpha \), i.e., \( 2\alpha \leq n\lceil \frac{k}{2} \rceil - \frac{k}{2} \). So, there is a path of length at most \( \lceil \frac{k}{2} \rceil + \frac{k}{2}\lceil \frac{k}{2} \rceil - \frac{1}{2}\lceil \frac{k}{2} \rceil \leq \frac{k}{2}(n + 1) + \frac{\alpha}{2} \) from \( v \) to \( u \). If \( 2\alpha > n\lceil \frac{k}{2} \rceil - \frac{k}{2} \) then there is a path of length less than \( n\lceil \frac{k}{2} \rceil - n\lceil \frac{k}{2} \rceil + \frac{k}{2}\lceil \frac{k}{2} \rceil \leq \frac{k}{2}(n + 1) + \frac{\alpha}{2} \). Thus, when \( k \) is odd there is a path of length at most \( \frac{k}{2}(n + 1) + \frac{\alpha}{2} \) from \( v \) to \( u \).

Note that we only have an upper bound on the diameter of \( AQ_{n,k} \), when \( n \geq 3 \). Ascertaining the exact value of the diameter appears to be combinatorially quite challenging. However, we conjecture that our upper bound is actually quite close to the true diameter.

### 5.6 Conclusions

In this chapter, we have defined a new class of graphs, the class of augmented \( k \)-ary \( n \)-cubes, and we have examined these graphs in relation to some properties pertinent to their use as interconnection networks for parallel computing. Let us examine our findings by comparing and contrasting augmented \( k \)-ary \( n \)-cubes with (the standard) \( k \)-ary \( n \)-cubes from which they are derived.

Both \( AQ_{n,k} \) and \( Q_n^k \) have \( k^n \) vertices, with the former having \( (n - 1)k^n \) more edges than the latter, and both interconnection networks are Cayley graphs, and so vertex-symmetric. However, \( AQ_{n,k} \) has a much improved connectivity of \( 4n - 2 \)
5.6. Conclusions

in comparison with the connectivity of $Q_n^k$ which is $2n$, although this comes at the expense of an increased vertex degree, which is $4n - 2$ as opposed to $2n$ for the $k$-ary $n$-cube (both $AQ_{n,k}$ and $Q_n^k$ are 'maximally connected', in the sense that if disjoint paths are used to transmit messages from one vertex to another in either network then there are no unused neighbours of the source vertex). We have also shown an upper bound on the diameter of an augmented $k$-ary $n$-cube at roughly one half that of a $k$-ary $n$-cube.

Recall that both the $k$-ary $n$-cube and the augmented $k$-ary $n$-cube come with two parameters which are both variable. Suppose that we have a $k$-ary $n$-cube, which involves $n^k$ vertices, and we wish to obtain an augmented $K$-ary $N$-cube of comparable size, but not necessarily by choosing the parameters $N = n$ and $K = k$, so that the degrees of the two networks are also comparable. Choose

$$N = \frac{n}{2} \text{ and } K = \frac{k}{1 - \frac{1}{\log(n)}}$$

(we assume for simplicity that both $N$ and $K$ are integral). Thus, $n^k = N^K$. Moreover, the degree of the $k$-ary $n$-cube $Q_n^k$ is $2n$ and the degree of the augmented $K$-ary $N$-cube $AQ_{N,K}$ is $2n - 2$, with the diameter of $Q_n^k$ being $\frac{n^k}{2}$ in comparison to an upper bound of

$$\frac{K}{4}(N + 1) = \frac{k}{4(1 - \frac{1}{\log(n)})}(\frac{n}{2} + 1)$$

on the diameter of $AQ_{N,K}$ (again, for notational simplicity, let us assume that $k$ is even). It is easy to see that for any fixed $k$, as $n$ increases the diameter of our augmented $K$-ary $N$-cube approaches one quarter of that of our $k$-ary $n$-cube (indeed, the actual improvement in diameter could well be better than this, given that we have only given an upper bound as to the diameter of a $AQ_{K,N}$). In consequence, we conclude that augmented $k$-ary $n$-cubes can be regarded as improvements over $k$-ary $n$-cubes.

There are numerous directions for further research. One obvious one is an exact characterization of the diameter of an augmented $k$-ary $n$-cube. However, even in the absence of this exact characterization, our upper bound results still yield a significant improvement. Moreover, the constructions used in the proof of Proposition 5.5.1 yield a very simple routing algorithm of time complexity $O(nk)$ (albeit possibly
5.6. Conclusions

The lengths of the longest of the $4n - 2$ disjoint paths constructed in $AQ_{n,k}$ in the proof of Theorem 5.4.3 is longer than the length of the $2n$ disjoint paths joining any two distinct vertices of $Q_n^k$ constructed in [45]; for in [45], $2n$ disjoint paths, joining any two distinct nodes $u$ and $v$ of $Q_n^k$, were constructed so that the lengths of these paths are 0, 2, or $4 + d_{Q_n^k}(u, v)$, except for one path in a special case (when the Hamming distance between the $u$ and $v$ is 1) where the length of the path might be $4 + d_{Q_n^k}(u, v)$. This is possibly to be expected, given that we are constructing $4n - 2$ paths in $AQ_{n,k}$ whereas only $2n$ paths were constructed in $Q_n^k$ in [45]. Nevertheless, it would be interesting to try and improve upon our length bounds.

Finally, there are numerous other aspects relating to augmented $k$-ary $n$-cubes which are worthy of study: for example, the embedding of other networks in $AQ_{n,k}$ (cf. [11,12,41]), the tolerance of faults within $AQ_{n,k}$ (cf. [11,14]), and broadcasting and routing in $AQ_{n,k}$ (cf. [13,41]).
Chapter 6

One-to-Many Node-Disjoint paths in \((n, k)\)-star graph

6.1 Introduction

Chiang and Chen [39] introduced \((n, k)\)-star graphs, \(S_{n,k}\), where \(n > k \geq 1\), as alternatives to \(n\)-star graphs, for which the 'jump' from \(n\) nodes in an \(n\)-star graph to \((n+1)\) nodes in an \((n+1)\)-star graph is deemed excessive (\(n\)-star graphs were devised in [4] as rivals to hypercubes in that they can incorporate comparable numbers of nodes yet have smaller diameters and degrees). The two parameters, \(n\) and \(k\), of \((n, k)\)-star graphs allow much more precision with regard to incorporating more nodes, and allow fine tuning with regard to a degree/diameter trade-off.

As regards the node-connectivity of \(S_{n,k}\), it was shown in [38] that there are \(n - 1\) node-disjoint paths joining any two distinct nodes of \(S_{n,k}\) (with an implicit algorithm for construction) and that each of these paths has length at most the diameter, \(\text{dia}(S_{n,k})\), of \(S_{n,k}\) plus 3. Furthermore, it was shown that the diameter \(\text{dia}(S_{n,k})\) is \(2k - 1\), if \(1 \leq k \leq \lfloor \frac{n}{2} \rfloor\), and \(k + \lfloor \frac{n-1}{2} \rfloor\), if \(\lfloor \frac{n}{2} \rfloor + 1 \leq k < n\). Thus, the one-to-one node-disjoint paths problem for \(S_{n,k}\) has been pretty much resolved (note that as \(S_{n,k}\) is regular of degree \(n - 1\), there is no scope for incorporating more node-disjoint paths between two nodes). In this chapter, we are concerned with the many-to-one node-disjoint paths problem for \(S_{n,k}\); that is, we are given in \(S_{n,k}\), \(n - 1\) distinct target nodes, in the set \(T\), and a source node \(I\), different from any target
node, and we wish to find \( n - 1 \) node disjoint paths, one from each target node of \( T \), to \( I \).

The many-to-one node-disjoint paths problem is a fundamental problem in the design and implementation of parallel and distributed computing systems and it has been extensively studied for a variety of (families of) interconnection networks. Whilst Menger's Theorem [21] implies that, given a source node and \( n - 1 \) distinct target nodes (different from the source) in a graph of node-connectivity \( (n-1) \), there exist \( n - 1 \) node-disjoint paths from each of the target nodes to the source, it is by no means easy to identify and actually construct the paths, especially if the paths are to be as short as possible. Indeed, given a source and a collection of target nodes in an arbitrary graph, the general problem of finding node-disjoint paths from each of the target nodes to the source with each path of shortest length is NP-hard [83]. However, in many interconnection networks, which almost always have 'uniformity' properties such as recursive decomposability, node-symmetry and degree regularity, the situation is much more acceptable (see, for example, [4, 20, 34, 63, 71, 74, 84, 102, 114, 129, 136]). We only highlight here two such studies of the many-to-one node-disjoint paths problem: in hypercubes and in n-star graphs. In [136], Rabin studied the many-to-one node-disjoint paths problem in hypercubes where he showed that given a source node and \( n \) target nodes in an \( n \)-dimensional hypercube, there exist node-disjoint paths from each of the target nodes to the source such that each path has length at most 1 plus the diameter of the \( n \)-dimensional hypercube (that is, \( n \)). In [71], Gu and Peng showed that given a source and \( n - 1 \) target nodes in an \( n \)-star graph, there is an algorithm of time complexity \( O(n^2) \) that builds \( n - 1 \) paths from each of the target nodes to the source such that the length of each path is at most the diameter of the \( n \)-star graph (that is, \( \lfloor \frac{3(n-1)}{2} \rfloor \) plus 2.

In this chapter, we prove the following theorem.

**Theorem 6.1.1** When \( T \) is a set of \( n - 1 \) distinct nodes in \( S_{n,k} \), where \( n > k \geq 1 \), and when \( I \) is a node not in \( T \), there is an algorithm which finds \( n - 1 \) node-disjoint paths in \( S_{n,k} \) from the nodes in \( T \) to the node \( I \). Furthermore, all paths found by this algorithm have length at most \( 6k - 7 \) and the time complexity of the algorithm is \( O(k^3n^4) \).
Compared to Chapter 3, 4 and 5, where structural results are given, in this chapter, we will give an algorithmic result. Our algorithmic result contains a structural result. It should be noted that the structural results from Chapters 3, 4 and 5 can be translated into algorithms.

We present the basic definitions in Section 6.2 before dealing with the case when \( k = 2 \) in Section 6.3. In Section 6.4, we present the algorithm alluded to in Theorem 6.1.1 and its proof of correctness, and in Section 6.5 we consider the lengths of the paths constructed by our algorithm and also the time complexity of our algorithm. Our conclusions are presented in Section 6.6.

### 6.2 Basic definitions and lemmas

It is worthwhile beginning with an \( n \)-star graph in order that we might understand why \( (n, k) \)-star graphs emerged. In order to avoid the significant jump from \( n! \) nodes in an \( n \)-star graph to \( (n + 1)! \) nodes in an \( (n + 1) \)-star graph, \( (n, k) \)-star graphs were devised, as 'generalized' \( n \)-star graphs. \( S_{n,k} \) has \( \frac{n!}{(n-k)!} \) nodes and \( \frac{n-1}{2} \times \frac{n!}{(n-k)!} \) edges. Note that \( S_{n,n-1} \) is isomorphic to the \( n \)-star \( S_n \), and that \( S_{n,1} \) is a clique on \( n \) nodes.

An important property of \( S_{n,k} \), which we make crucial use of, is that it can be partitioned into \( n \) node-disjoint copies of \( S_{n-1,k-1} \) over one of \( k - 1 \) dimensions. In more detail, let \( i \in \{2, 3, \ldots, k\} \) and partition the nodes of \( S_{n,k} \) by fixing the \( i \)th component of each node. Thus, define \( S'_{n,k}(j) = \{(u_1, u_2, \ldots, u_k) \in V(S_{n,k}) : u_i = j\} \), for each \( j \in \{1, 2, \ldots, n\} \). It is trivial to see that the set of nodes \( S'_{n,k}(j) \), for \( j \in \{1, 2, \ldots, n\} \), induces a copy of \( S_{n-1,k-1} \). Note that there are \( k - 1 \) dimensions over which we can so partition \( S_{n,k} \).

We adopt the following notation throughout this chapter. Let \( I = (u_1, u_2, \ldots, u_k) \) be an arbitrary node of \( S_{n,k} \). Note that there are \( k - 1 \) neighbours of \( I \) that are joined to \( I \) via an \( i \)-edge, and \( n - k \) neighbours of \( I \) that are joined to \( I \) by a 1-edge; each neighbour is characterized by its first component. We denote the neighbour of \( I \) whose first component is \( j \) by \( I^j \). We shall denote paths in \( S_{n,k} \) by \( \rho(t, s) \) where \( t \) is the start node and \( s \) is the terminal node. Paths are written explicitly as sequences of nodes, such as \( (t, u_2, u_3, \ldots, u_m, s) \). We write \( x \in S_{n,k} \setminus T \), where \( T \) is a set of
nodes of $S_{n,k}$, to denote that $x$ is a node of $S_{n,k}$ different from any node in $T$.

Our intention is to build an algorithm to find $n - 1$ node-disjoint paths from each of $n - 1$ distinct target nodes, held in $T$, to a given source node $I$ of $S_{n,k}$ ($I$ is never a target node). Before we present our algorithm, we show that there are certain assumptions that we can make.

Lemma 6.2.1 Let $T$ be a set of $n - 1$ target nodes in $S_{n,k}$, where $k \geq 3$. There exists a dimension $i \in \{2, 3, \ldots, k\}$ such that each of $S_{n,k}^i(1), S_{n,k}^i(2), \ldots, S_{n,k}^i(n)$ contains at most $n - 2$ nodes of $T$.

Proof: Suppose that for every $j \in \{2, 3, \ldots, k\}$, when we partition $S_{n,k}$ over dimension $j$, we get that some $S_{n,k}^j(i_j)$ contains all the target nodes from $T$. Thus, all target nodes in $T$ have the form $(u, i_2, i_3, \ldots, i_k)$, for some $u$. This yields a contradiction as there are only $n - (k - 1)$ such nodes. 

Suppose that $k \geq 3$. By Lemma 6.2.1, we can choose a dimension, $i$, say (where $i \in \{2, 3, \ldots, k\}$), so that when we partition the $(n, k)$-star $S_{n,k}$ over dimension $i$ to obtain the $(n - 1, k - 1)$-stars $S_{n,k}^i(1), S_{n,k}^i(2), \ldots, S_{n,k}^i(n)$, we can be sure that each $S_{n,k}^i(j)$ contains at most $n - 2$ target nodes. Suppose that $i \neq k$. The automorphism of $S_{n,k}$ obtained by swapping the $i$th and $k$th components of every node is such that $S_{n,k}^i(j)$ is mapped to $S_{n,k}^k(j)$. Suppose that $I = (y_1, y_2, \ldots, y_k)$ and let $\sigma$ be any permutation of $\{1, 2, \ldots, n\}$ for which $\sigma(y_j) = j$, for $j = 1, 2, \ldots, k$. The permutation $\sigma$ yields an automorphism of $S_{n,k}$ by mapping each node $(x_1, x_2, \ldots, x_k)$ to $(\sigma(x_1), \sigma(x_2), \ldots, \sigma(x_k))$, so that each $S_{n,k}^k(j)$ is mapped to $S_{n,k}^k(\sigma(j))$. Thus, we may assume that our source node $I$ is $I_k = (1, 2, \ldots, k)$ and that when we partition over dimension $k$, the resulting $(n - 1, k - 1)$-stars $S_{n,k}^k(1), S_{n,k}^k(2), \ldots, S_{n,k}^k(n)$ each contains at most $n - 2$ target nodes. Note that when $k = 2$, we can assume that our source is $I_k$ but not that partitioning over dimension $k$ results in $(n - 1, k - 1)$-stars each containing at most $n - 2$ target nodes. Henceforth, for brevity, we denote $S_{n,k}^k(i)$ by $S_i$ (with $S_i$ not to be confused with the $n$-star graph of the same name).

For $i \in \{k + 1, k + 2, \ldots, n\}$, we define $I_i = (k, 2, 3, \ldots, k - 1, i) \in S_i$; for $i \in \{2, 3, \ldots, k - 1\}$, we define $I_i = (k, 2, 3, \ldots, i - 1, 1, i + 1, \ldots, k - 1, i) \in S_i$; and we define $I_1 = (k, 2, 3, \ldots, k - 1, 1) \in S_1$. For $i = 1, 2, \ldots, n$, we denote the set of
target nodes of $T$ which lie in $S_i$, that is, $T \cap S_i$, by $T_i$.

6.3 The case for $k = 2$

In this section, we devise an algorithm `Disjoint_paths_when_k=2 (S_{n,2}, T, I_2, paths)` which finds node-disjoint paths in $S_{n,2}$ from $n - 1$ target nodes in $T$ to the source node $I_2$ (which is not a target node); the paths are returned in `paths`. (Note that the many-to-one node-disjoint paths problem is trivial for $S_{n,1}$, an $n$-clique.)

As is the case throughout, it is best to study the algorithm in conjunction with the subsequent description.

1. `Disjoint_paths_when_k=2 (S_{n,2}, T, I_2, paths)`
2. for every node $I_2^j \in T_2$ do
3. add the path $\rho(I_2^j, I_2) = (I_2^j, I_2)$ to `paths`;
4. od
5. set `free` := \{ $S_j : j \in \{1, 3, 4, \ldots, n\}$ and $T_j = \emptyset$, and if $j \neq 1$ then $I_2^j \notin T_2$\};
6. for $i = 1, 2, \ldots, n$ where $i \neq 2$ and $T_i \neq \emptyset$ do
7. if $i = 1$ or $I_2^i \notin T_2$ then
8. if $I_i \in T_i$ then
9. add the path $\rho(I_i, I_2) = (I_i, I_2)$ (resp. $\rho(I_i, I_2) = (I_i, I_2^i, I_2)$) to `paths` if $i = 1$ (resp. $i \neq 1$);
10. `sorted_target` := $I_i$;
11. else
12. choose some $I_i^j \in T_i$ and add the path $\rho(I_i^j, I_2) = (I_i^j, I_1, I_2)$ (resp. $\rho(I_i^j, I_2) = (I_i^j, I_i, I_2^i, I_2)$) to `paths` if $i = 1$ (resp. $i \neq 1$);
13. `sorted_target` := $I_i^j$;
14. fi
15. else
16. `sorted_target` := $\epsilon$;
17. fi
6.3. The case for $k = 2$

18 if $\text{sorted.target} \neq \epsilon$ then
19 \hspace{1em} let $\text{good.free} \subseteq \text{free}$ be of size $|T_i| - 1$;
20 \hspace{1em} else
21 \hspace{2em} let $\text{good.free} \subseteq \text{free}$ be of size $|T_i|$;
22 \hspace{1em} fi
23 \hspace{1em} $\text{free} := \text{free} \setminus \text{good.free}$;
24 \hspace{1em} for every $I_i \in T_i \setminus \{\text{sorted.target}\}$ do
25 \hspace{2em} if $S_j \in \text{good.free}$ then
26 \hspace{3em} add the path $\rho(I_i, I_2) = (I_i^1, I_i^1, I_1, I_2)$ (resp. $\rho(I_i, I_2) = (I_i^1, I_i^2, I_j, I_2)$) to paths if $j = 1$ (resp. $j \neq 1$);
27 \hspace{3em} remove $S_j$ from $\text{good.free}$;
28 \hspace{2em} else
29 \hspace{3em} choose $I_i \not\in T_i$ for which $S_i \in \text{good.free}$;
30 \hspace{3em} add the path $\rho(I_i, I_2) = (I_i^2, I_i^1, I_1, I_2)$ (resp. $\rho(I_i, I_2) = (I_i^2, I_i^1, I_1, I_2)$) to paths if $l = 1$ (resp. $l \neq 1$);
31 \hspace{3em} remove $S_i$ from $\text{good.free}$;
32 \hspace{2em} fi
33 \hspace{1em} od
34 od

(We remark that with respect to line 5, and elsewhere throughout, when we say that, for example, $S_3$ is in the set $\text{free}$, in any implementation we would simply hold the index 3 in $\text{free}$; we write it as we do to make our algorithm more understandable.)

The actions of $\text{Disjoint.paths.when.k=2}$ can be described as follows. In lines 2-4, we define paths from every target node in $S_2$ to $I_2$. In line 5, we define $\text{free}$ to consist of those $S_j$'s from $\{S_1, S_2, \ldots, S_n\} \setminus \{S_2\}$ containing no target nodes and for which the node $I_2 \not\subseteq T_2$ (if $j \neq 1$); some of these $S_j$'s will be used as collections of 'transit' nodes for paths from target nodes (in other $S_i$'s) to $I_2$.

In lines 6-34, we deal with the $S_i$'s for which $i \neq 2$ and $T_i \neq \emptyset$ in turn. In lines 7-17, we ensure that if $I_2 \not\subseteq T_2$, i.e., $I_2$ does not block a path from $I_i$ to $I_2$ through $I_1$, or $i = 1$ then a path from one of the target nodes in $T_i$ through $I_i$ to $I_2$ is chosen. The target in $T_i$ chosen is registered in $\text{sorted.target}$.
In lines 18-22, a subset \texttt{good.free} of \texttt{free} of size $|T'_i| - 1$ is chosen, if \texttt{sorted.target} exists, and of size $|T'_i|$ otherwise. We need to verify that such a subset exists. Suppose that $X = \{l : l = 1, 3, 4, \ldots, n, l < i, T_i \neq \emptyset\}$ with $Y \subseteq X$ defined as $Y = \{l : l \in X \setminus \{1\}, I'_2 \in T_2\}$, i.e., $X$ indexes the $S_i$'s that have so far been dealt with in the for-loop in lines 6-34, and $Y$ indexes those such $S$'s for which $I'_2$ blocks direct paths from $I_l$ to $I_2$. On an iteration of the for-loop for some $i$ where $i \neq 2$ and $T_i \neq \emptyset$, any $S_i$ from $\{S_1, S_2, \ldots, S_n\} \setminus \{S_2, S_i\}$ fails to be in \texttt{free} for exactly one of six reasons:

1. \( l \in Y \);
2. \( S_l \) is used as a set of transit nodes for a path from some target in $S_j$ where \( j \in Y \);
3. \( l \in X \setminus Y \);
4. \( S_l \) is used as a set of transit nodes for a path from some target in $S_j$ where \( j \in X \setminus Y \);
5. \( l \notin X, l \neq 1 \) and \( I'_2 \in T_2 \); and
6. \( l \notin X, (I'_2 \notin T_2 \text{ or } l = 1) \) and $T_i \neq \emptyset$.

Some of the different cases are illustrated in Fig. 6.1, where the target nodes are represented in black and where $i = 18$ (note that all $S_j$'s are cliques even though they are not depicted as such). We can associate a target node with any $S_i$ in \texttt{free} by choosing: the target node $I'_2$ in case 1; the unique target node $t$ upon whose path $\rho(t, I_2)$ the nodes of $S_l$ are used as transit nodes in cases 2 and 4; the target node $t$ of $S_l$ for which the path $\rho(t, I_2)$ passes through $I_l$ in case 3; the target node $I'_2$ in case 5; and any target node of $T_i$ in case 6. All such target nodes are distinct and are different from the target nodes in $T_i$. Thus, $|\texttt{free}| \geq (n - 2) - ((n - 1) - |T_i|) = |T_i| - 1$. Furthermore, if \texttt{sorted.target} = $\epsilon$ then $I'_2 \in T_2$ and $i \neq 1$, and this target node is distinct from all target nodes which were associated above; hence, $|\texttt{free}| \geq (n - 2) - ((n - 1) - |T_i| - 1) = |T_i|$ and our claim holds.
In line 23, we remove the copies of $S_{n-1,1}$ that are in \texttt{good} from \texttt{free}.

In lines 24-33, we deal with the target nodes of $T_i$ in turn and build paths to $I_2$. This is done as follows. If $I^j_i \in T_i \setminus \{\text{\texttt{sorted_target}}\}$ and $S_j \in \texttt{good}$ then we simply take the path from $I^j_i$ through $S_j$ and on to $I_2$; otherwise, if $I^j_i \not\in T_i \setminus \{\text{\texttt{sorted_target}}\}$ and $S_j \notin \texttt{good}$ then we choose a neighbour $I^j_i$ of $I^j_i$ in $S_i$ that is not a target and where $S_i \in \texttt{good}$ (such a neighbour always exists because we have chosen \texttt{good} large enough and $S_{n-1,1}$ is a clique). Consequently, \texttt{Disjoint_paths} when $\chi=2$ achieves its aims. Furthermore, all paths found by \texttt{Disjoint_paths} when $\chi=2$ have length at most 5 and the time complexity of \texttt{Disjoint_paths} when $\chi=2$ is $O(n^2)$.

**Theorem 6.3.1** When $T$ is a set of $n - 1$ distinct nodes in $S_{n,2}$ and when $I$ is a node not in $T$, the algorithm \texttt{Disjoint_paths} when $\chi=2(S_{n,2}, T, I, \text{\texttt{paths}})$ finds $n - 1$ node-disjoint paths from the nodes in $T$ to the node $I$. Furthermore, all paths found have length at most 5 and the time complexity of \texttt{Disjoint_paths} when $\chi=2$ is $O(n^2)$.

### 6.4 Building node-disjoint paths

We now detail a recursive algorithm \texttt{Disjoint_paths}($S_{n,k}, T, I_k, \text{\texttt{paths}}$) to construct node-disjoint paths from $n - 1$ distinct target nodes in $S_{n,k}$, given by the set of nodes $T$, to a source node $I_k$ (which is different from each target node). The $n - 1$ paths will be returned in \texttt{paths}.
6.4.1 The basic algorithm

Roughly speaking, our algorithm \texttt{Disjoint\_paths} proceeds as follows. First, we find disjoint paths from the target nodes in \( S_k \) to \( I_k \) (if any such target nodes exist); these paths are not changed throughout the subsequent execution of the algorithm. A neighbour \( I_k \) of \( I_k \) appearing on one of these paths cannot be used in another path from \( I_i \) and so 'blocks' \( S_i \); consequently, the set \textit{blocked} consists of those \( S_i \)'s that are blocked by some neighbour of \( I_k \) in \( S_k \). Next, we deal in turn with the \( S_i \)'s for which \( T_i \neq \emptyset \). Once the paths from the target nodes of such an \( S_i \) to \( I_k \) have been established, they do not change throughout the subsequent execution of the algorithm. Our basic algorithm is as follows. In the rest of this section, we detail the procedures in the algorithm and prove that our algorithm is correct.

```plaintext
1   Disjoint\_paths\( (S_{n,k}, T, I_k, paths) \)
2   if \( k = 2 \) then
3       call Disjoint\_paths\_when\_k=2\( (S_{n,2}, T, I_2, paths) \);
4   else
5       free := \{S_j : j = 1,2,\ldots,n, j \neq k, T_j = \emptyset\};
6       some := \{S_j : j = 1,2,\ldots,n, j \neq k, T_j \neq \emptyset\};
7       paths := \emptyset;
8       blocked := \emptyset;
9       used := \emptyset;
10      if \( T_k \neq \emptyset \) then
11          call Paths\_in\_S\( \_k (S_k, T_k, I_k, free, paths, blocked) \);
12      fi
13      if there is some \( S_{i_0} \in \text{blocked} \cap \text{some} \) then
14          if \( I_i \in T_i \) then
15             call Paths\_in\_some\_target\_and\_blocked\( (S_{i_0}, T_{i_0}, I_{i_0}, free, used, paths) \);
16          else
17             call Paths\_in\_some\_not\_target\_and\_blocked\( (S_{i_0}, T_{i_0}, I_{i_0},
free, used, paths) \);
18          fi
19      fi
```
We have one remark concerning procedure calls in our algorithm (including the procedures to follow). As Lemma 6.2.1 shows, we can always assume that when dealing with $S_{n,k}$, our source is $I_k$ and, for $k \geq 3$, when we partition over dimension $k$, none of the resulting copies of $S_{n-1,k-1}$ contains more than $n - 2$ target nodes. As can be seen from the above outline algorithm (in conjunction with a closer look at the procedures to follow), we make a number of recursive calls to Disjoint_paths. Even though we do not explicitly state this in the procedures to follow, we always assume that we have arranged things (using automorphisms as in Section 6.2) so that if in some recursive call we are dealing with a copy of $S_{n',k'}$ then our source is $I_{k'}$ and none of the copies of $S_{n'-1,k'-1}$ resulting from partitioning over dimension $k'$ contains more than $n' - 2$ target nodes. From Lemma 6.2.1, we can decide which automorphisms (and their inverses) to apply and applying these automorphisms, and ensuring that we partition over dimension $k$ (in $S_{n,k}$, to get at most $n - 2$ target nodes in each resulting $S_i$) by checking if all the target nodes in one subgraph for each dimension. There are at most $k - 1$ dimensions and $n - 2$ target nodes to be considered. To apply the automorphism (mapping) will only take $O(n)$ time (suppose we choose a mapping function $f(i) = j$, we do not need to apply this on every node, but in the consequent processing, we will run the mapping function to the corresponding node first). Hence, this can clearly be done in $O(kn)$ time.

### 6.4.2 Paths in $S_k$

We start with $\text{Paths.in.} \ S_k(S_k, T_k, I_k, free, blocked, paths)$, which returns a set $free$ of $S_i$'s, a set $paths$ of paths in $S_k$ and a set $blocked$ of $S_i$'s, where $k \geq 3$. 
Pre-conditions assumed by this algorithm are that \( \text{free} = \{S_j : j = 1, 2, \ldots, n, j \neq k, T_j = \emptyset\} \), \( \text{paths} = \emptyset \), \( \text{blocked} = \emptyset \) and \( 0 < |T_k| < n - 1 \).

1. \( \text{Paths} \_\text{in} \_S_k(S_k, T_k, I_k, \text{free}, \text{paths}, \text{blocked}) \)
2. \( \text{temp} := \emptyset; \)
3. \( \text{for each neighbour } I^j_k \text{ of } I_k \text{ in } S_k \setminus T_k \text{ do} \)
4. \( \text{if } S_j \notin \text{free} \text{ and } |\text{temp} \cup T_k| < n - 2 \text{ then} \)
5. \( \text{add } I^j_k \text{ to } \text{temp}; \)
6. \( \text{fi} \)
7. \( \text{od} \)
8. \( \text{temp} \_\text{so} \_\text{far} := \text{temp}; \)
9. \( \text{for each neighbour } I^j_k \text{ of } I_k \text{ in } S_k \setminus (\text{temp} \_\text{so} \_\text{far} \cup T_k) \text{ do} \)
10. \( \text{if } |\text{temp} \cup T_k| < n - 2 \text{ then} \)
11. \( \text{add } I^j_k \text{ to } \text{temp}; \)
12. \( \text{fi} \)
13. \( \text{od} \)
14. \( \text{call } \text{Disjoint} \_\text{paths}(S_k, \text{temp} \cup T_k, I_k, \text{paths}); \)
15. \( \text{for each neighbour } I^j_k \text{ of } I_k \text{ in } S_k \text{ do} \)
16. \( \text{if } I^j_k \in \text{temp} \text{ then} \)
17. \( \text{remove } \rho(I^j_k, I_k) \text{ from } \text{paths}; \)
18. \( \text{else} \)
19. \( \text{add } S_j \text{ to } \text{blocked}; \)
20. \( \text{if } S_j \in \text{free} \text{ then} \)
21. \( \text{remove } S_j \text{ from } \text{free}; \)
22. \( \text{fi} \)
23. \( \text{fi} \)
24. \( \text{od} \)

The actions of \( \text{Paths} \_\text{in} \_S_k \) can be described as follows. Initially, \( \text{free} \) consists of the \( S_i \)'s from \( \{S_1, S_2, \ldots, S_n\} \setminus \{S_k\} \) containing no target nodes; some of these \( S_i \)'s will be used as collections of 'transit' nodes for paths from target nodes (in other \( S_j \)'s) to \( I_k \). In line 2, \( \text{temp} \) is initialized as an empty set of nodes. In lines
3-13, some of the neighbours of $I_k$ in $S_k$ are then set as temporary target nodes so that $S_k$ has exactly $n - 2$ (the degree of $I_k$ in $S_k$) target nodes and temporary target nodes. The order in which the neighbours of $I_k$ are chosen to be temporary target nodes is important; those neighbours $I_k^j$ for which $S_j$ contains at least one target node are chosen first, before neighbours $I_k^{j'}$ for which $S_{j'}$ contains no target nodes are chosen (as to whether neighbours $I_k^{j'}$ for which $S_{j'}$ contains no target nodes are chosen depends upon the distribution of the target nodes).

In line 14, node-disjoint paths are recursively constructed from these target nodes and temporary target nodes to $I_k$ (note that there are $n - 2$ of these paths and that every neighbour of $I_k$ in $S_k$ lies on exactly one of these paths). In line 17, the paths involving temporary target nodes are then removed from paths (note that these paths are just solitary edges). The remaining paths, from target nodes in $S_k$ to $I_k$, will end up being output by the algorithm, and in line 19 the $S_j$'s for which $I_k^j$ lies on one of these paths are registered in blocked (so, any node $I_k^j$ of $S_k$ for which $S_j \in blocked$ cannot be used on any path from the node $I_j$ in $S_j$ to $I_k$). Finally, those $S_j$'s in $blocked \cap free$ are removed from $free$ as they can no longer be used as collections of transit nodes, since there is no path to $I_k$ from $I_j$ through $I_k^j$ (or directly, if $j = 1$). Note that the total number of temporary target nodes chosen is $(n - 2) - |T_k|$ and that the total number of $S_j$'s from $\{S_1, S_2, \ldots, S_n\} \setminus \{S_k\}$ containing at least one target node is at most $(n - 1) - |T_k|$. Thus, after execution of Paths.in.S_k there is at most one $S_j$ for which $S_j \in blocked$ and $S_j \in some$ (some is fixed throughout at those $S_i$'s for which $T_i \neq \emptyset$).

6.4.3 Paths in $S_i \in blocked \cap some$

Suppose that $S_i$ is such that $S_i \in blocked \cap some$. Suppose further that $I_i \in T_i$. The procedure Paths.in.some.target_and_blocked($S_i, T_i, I_i, free, used, paths$) finds paths to $I_k$ from every target node in $S_i$. Pre-conditions are that $I_i \in T_i$, $0 < |T_i| < n - 1$ and $used = \emptyset$.

1 Paths.in.some.target_and_blocked($S_i, T_i, I_i, free, used, paths$)
2 choose some neighbour $I_i^j$ of $I_i$ in $S_i$ for
which \( I_i^j \notin T_i \) and \( S_j \in \text{free} \);

3. \( \text{root.escape} := I_i^j \);

4. \( \text{temp} := \{\text{root.escape}\} \);

5. for each neighbour \( I_i^j \) of \( I_i \) in \( S_i \setminus (\{\text{root.escape}\} \cup T_i) \) do

6. if \( S_j \notin \text{free} \) and \( |\text{temp} \cup (T_i \setminus \{I_i\})| < n - 2 \) then

7. add \( I_i^j \) to \( \text{temp} \);

8. fi

9. od

10. \( \text{temp.so.far} := \text{temp} \);

11. for each neighbour \( I_i^j \) of \( I_i \) in \( S_i \setminus (\text{temp.so.far} \cup T_i) \) do

12. if \( |\text{temp} \cup (T_i \setminus \{I_i\})| < n - 2 \) then

13. add \( I_i^j \) to \( \text{temp} \);

14. fi

15. od

16. call Disjoint_paths(\( S_i, (\text{temp} \cup T_i) \setminus \{I_i\}, I_i, S_i\_paths \));

17. \( S_i\_paths\_blocked := \emptyset \);

18. for every neighbour \( I_i^j \) of \( I_i \) in \( S_i \) do

19. case of

20. \( I_i^j = \text{root.escape} \):

21. replace the path \( \rho(\text{root.escape}, I_i) \) in \( S_i\_paths \)

with the path \( \rho(I_i, \text{root.escape}) = (I_i, \text{root.escape}) \);

22. set \( \text{escape}[\rho(I_i, \text{root.escape})] := S_j \);

23. remove \( S_j \) from \( \text{free} \) and add \( S_j \) to \( \text{used} \);

24. \( I_i^j \in \text{temp} \setminus \{\text{root.escape}\} \):

25. remove the path \( \rho(I_i^j, I_i) \) from \( S_i\_paths \);

26. \( I_i^j \notin \text{temp} \cup T_i \):

27. replace the path \( \rho(t, I_i) \) in \( S_i\_paths \)

upon which \( I_i^j \) lies with the sub-path \( \rho(t, I_i^j) \);

28. set \( \text{escape}[\rho(t, I_i^j)] := S_j \);

29. remove \( S_j \) from \( \text{free} \) and add \( S_j \) to \( \text{used} \);

30. \( I_i^j \in T_i \) and \( S_j \in \text{free} \):
replace the path \( p(I_i^j, I_l^j) \) in \( S_i\text{-paths} \)
with the path \( p(I_i^j, I_l^j) = (I_i^j) \);
set \( \text{escape}[p(I_i^j, I_l^j)] := S_j \);
remove \( S_j \) from \( \text{free} \) and add \( S_j \) to \( \text{used} \);
\( I_i^j \in T_i \) and \( S_j \notin \text{free} : \)
remove the path \( p(I_i^j, I_l^j) \) from \( S_i\text{-paths} \) and
add the path \( p(I_i^j, I_l^j) = (I_i^j) \) to \( S_i\text{-paths}\text{-blocked} \);
esac;
while some path \( p(t, s) \) in \( S_i\text{-paths} \) contains a node whose
first component \( j \), say, is such that \( S_j \in \text{free} \) do
replace the path \( p(t, s) \) in \( S_i\text{-paths} \) with it’s sub-path \( p(t, x) \)
where \( x \) is such that its first component \( j \), say, is such
that \( S_j \in \text{free} \) and where if \( y \neq x \) is any other node on \( p(t, x) \)
then its first component \( j' \), say, is such that \( S_j' \notin \text{free} \);
remove \( \text{escape}[p(t, s)] \) from \( \text{used} \) and add \( \text{escape}[p(t, s)] \) to \( \text{free} \);
set \( \text{escape}[p(t, x)] := S_j \);
add \( S_j \) to \( \text{used} \) and remove \( S_j \) from \( \text{free} \);
od
for every path \( p(I_i^j, I_l^j) \in S_i\text{-paths}\text{-blocked} \) do
if \( S_j \in \text{free} \) then
remove \( p(I_i^j, I_l^j) \) from \( S_i\text{-paths}\text{-blocked} \)
and add \( p(I_i^j, I_l^j) \) to \( S_i\text{-paths} \);
set \( \text{escape}[p(I_i^j, I_l^j)] := S_j \);
remove \( S_j \) from \( \text{free} \) and add \( S_j \) to \( \text{used} \);
fi
od
for every path \( p(I_i^j, I_l^j) \in S_i\text{-paths}\text{-blocked} \) do
choose a neighbour \( (I_i^j)' \) of \( I_i^j \) in \( S_i \) for which \( S_i \in \text{free} \);
remove \( p(I_i^j, I_l^j) \) from \( S_i\text{-paths}\text{-blocked} \) and
add \( p(I_i^j, (I_l^j)') = (I_i^j, (I_l^j)') \) to \( S_i\text{-paths} \);
6.4. Building node-disjoint paths

71 set escape[\rho(I_i^j, (I_i^j)) := S_i;
72 remove S_i from free and add S_i to used;
73 od
74 for every path \rho(t,s) in S_i-paths do
75 extend \rho(t,s) to a path \rho(t,I_k) through the nodes of
76 S_j = escape[\rho(t,s)] to I_j and then on to I_k;
77 remove \rho(t,s) from S_i-paths and add \rho(t,I_k) to paths;
78 od

We explain below what Paths_in_some:target_and_blocked does, and prove that what the procedure claims to do is actually possible and that it achieves its aims.

We begin, in lines 2-3, by choosing a neighbour root.escape = I_i^j of I_i that is not a target node and through which a path from the target node I_j will pass on its way to I_k. We need to verify that there does indeed exist such a neighbour I_i^j. Suppose that when we attempt to choose our neighbour root.escape of I_i in S_i, we find that every neighbour I_i^j of I_i in S_i is such that S_j \not\in free. The reason any S_j \not\in free is that exactly one of the following holds: S_j \in blocked; S_j \in some \setminus blocked. Whatever the reason, we can associate a target node with I_i^j: if S_j \in blocked then choose the target node of T_k on whose path in paths the (blocking) node I_k lies; otherwise, if S_j \in some \setminus blocked then choose some target node of T_j (which is non-empty).

Note that it is never the case that two target nodes associated with two distinct neighbours of I_i in S_i are identical. Thus we get a contradiction as we obtain n - 2 distinct target nodes (corresponding to the n - 2 neighbours of I_i in S_i) and we have yet to consider the target node and the target node on whose path in S_k the node I_k lies.

The neighbour root.escape is set as a temporary target node in line 4. In lines 5-15, more neighbours of I_i are set as temporary target nodes, making sure that neighbours I_i^j for which S_j \not\in free are chosen before neighbours I_i^j for which S_j \in free. The process stops when |T_i| - 1 plus the number of temporary target nodes is exactly n - 2. We claim that all neighbours I_i^j of I_i that are not target nodes and for which S_j \not\in free are chosen as temporary target nodes. Let us count the
6.4. Building node-disjoint paths

number of \( S_i \)'s, from \( \{S_1, S_2, \ldots, S_n\} \setminus \{S_k, S_j\} \), that are not in \( \text{free} \). As above, the reason any \( S_i \not\in \text{free} \) is that exactly one of the following holds: \( S_i \in \text{blocked}; S_i \in \text{some} \setminus \text{blocked} \). Just as we did above, we can associate a target node with each such \( S_i \) so that distinct \( S_i \)'s are associated with distinct target nodes. Thus, the number of \( S_i \)'s, from \( \{S_1, S_2, \ldots, S_n\} \setminus \{S_k, S_j\} \), that are not in \( \text{free} \) is at most the number of target nodes that potentially can be associated with such an \( S_i \). This number is \((n - 1) - |T_i| - 1\) (as, by definition of how we associate target nodes, no target node in \( T_i \) can be associated with such an \( S_i \), and nor can the target node on whose path in \( S_k \) the node \( S_j \) lies). Thus, the number of temporary target nodes chosen, namely \((n - 2) - (|T_i| - 1) = (n - 1) - |T_i|\), is greater than \((n - 1) - |T_i| - 1\), which is no less than the number of \( S_i \)'s, from \( \{S_1, S_2, \ldots, S_n\} \setminus \{S_k, S_j\} \), that are not in \( \text{free} \). Thus, all neighbours \( I_i^j \) of \( I_i \) that are not target nodes and for which \( S_j \not\in \text{free} \) are chosen as temporary target nodes, with the consequence that any neighbour \( I_i^j \) of \( I_i \) that is neither a target node nor a temporary target node is such that \( S_j \in \text{free} \).

In line 16, we recursively find node-disjoint paths in \( S_i \) from every target node of \( T_i \setminus \{I_i\} \) and every temporary target node to the node \( I_i \). Note that all such paths from temporary target nodes to \( I_i \) necessarily consist of a single edge (as do such paths from target nodes that are neighbours of \( I_i \)) and that every neighbour of \( I_i \) in \( S_i \) lies upon exactly one such path. The paths reside in \( S_i \).paths.

In line 17, we initialize \( S_i \).paths.blocked as empty. In lines 18-37, we amend each path in \( S_i \).paths by working through the neighbours \( I_i^j \) of \( I_i \) in turn as follows. If \( I_i^j = \text{root.escape} \) then we amend the unique path containing \( I_i^j \) to \( \rho(I_i, \text{root.escape}) = (I_i, \text{root.escape}) \) and register that the nodes of \( S_j \) are to be used as transit nodes to extend \( \rho(I_i, \text{root.escape}) \) to a path to \( I_k \) and so can no longer be used as such for any other path (the nodes of \( S_j \) are available for this by choice of \( \text{root.escape} \)). This registration is done with the array \( \text{escape} \), indexed by our paths, and the set \( \text{used} \). If \( I_i^j \in \text{temp} \setminus \{\text{root.escape}\} \) then we simply remove the path \( \rho(I_i^j, I_i) \) from \( S_i \).paths. Otherwise, we truncate the unique path \( \rho(t, I_i) \) containing \( I_i^j \) by removing the final edge. Furthermore, if \( I_i^j \not\in T_i \) or \( S_j \in \text{free} \) then we register that the nodes of \( S_j \) are to be used as transit nodes for this (truncated) path (note that immediately...
after the recursive call, all neighbours \( I_i^j \) of \( I_i \) in \( S_i \) that are not target nodes nor temporary target nodes are such that \( S_j \in \text{free} \), and if \( I_i^j \in T_i \) and \( S_j \notin \text{free} \) then we move the path \( \rho(I_i^j, I_i^j) \) to the set \( S_i\text{-pathsblocked} \). Consequently, we have essentially dealt with every target node in \( S_i \) except possibly for some target nodes that are neighbours \( I_i^j \) of \( I_i \) in \( S_i \) where \( S_j \notin \text{free} \) (corresponding to the paths in \( S_i\text{-pathsblocked} \)).

In lines 38-43, we amend the paths of \( S_i\text{-paths} \) (remember, these are the paths from target nodes in \( S_i \) that can be trivially extended to paths to \( I_k \), through sets of transit nodes). Suppose that some path \( \rho(t, s) \) in \( S_i\text{-paths} \) is such that there is some node \( x \) of the form \((j, \ldots, i)\) lying upon it so that \( S_j \notin \text{free} \). We can replace the path \( \rho(t, s) \) in \( S_i\text{-paths} \) with the sub-path \( \rho(t, x) \), so long as we release the set of transit nodes \( \text{escape}[^{\rho(t, s)}] \) (for possible future use) and register that the new set of transit nodes \( S_j \) is not to be used as a set of transit nodes for any other path. By iterating this process, we get to the situation where no path in \( S_i\text{-paths} \) contains a node of the form \((j, \ldots, i)\) so that \( S_j \notin \text{free} \).

In lines 44-50, we deal with some of the paths in \( S_i\text{-pathsblocked} \), each of which is of the form \( \rho(I_i^j, I_i^j) \). The changes made in lines 38-43 might mean that \( S_j \) is now in \text{free} \), for such a path \( \rho(I_i^j, I_i^j) \); if so then we move \( \rho(I_i^j, I_i^j) \) to \( S_i\text{-paths} \) and register that the nodes of \( S_j \) are to be used as transit nodes to extend \( \rho(I_i^j, I_i^j) \) to a path to \( I_k \) and so can no longer be used as such for any other path.

In lines 51-56, we deal with the remaining paths in \( S_i\text{-pathsblocked} \) (of the form \( \rho(I_i^j, I_i^j) \) and where \( S_j \notin \text{free} \)). The situation can be visualized in Fig.6.2, where the target nodes are depicted in black, those paths \( \rho \) already established are depicted with an arrow (to \text{escape}[\rho] \), and the neighbours of \( I_i^j \) in \( S_i \setminus \{I_i\} \) are shaded in grey. We claim that there exists a neighbour \((I_i^j)^i \) of \( I_i^j \) in \( S_i \setminus \{I_i\} \) such that \( S_i \in \text{free} \). Let us count the number of \( S_i \)'s, from \( \{S_1, S_2, \ldots, S_n\} \setminus \{S_k, S_i\} \), for which \( S_i \notin \text{free} \). As we have seen already, any such \( S_i \) can be associated with a target node and all these associated target nodes are distinct. The maximum number of target nodes eligible to be associated with such an \( S_i \) is \((n - 1) - \alpha - 1\), where \( \alpha \) is the number of paths currently in \( S_i\text{-pathsblocked} \) (remember, the target node on whose path in \( S_k \) the node \( I_k^j \) lies is not eligible for association). Hence,
the number of $S_i$'s, from $\{S_1, S_2, \ldots, S_n\} \setminus \{S_k, S_i\}$, for which $S_i \in \text{free}$ is at least $(n - 2) - ((n - 1) - \alpha - 1) = \alpha \geq 1$. Consider the neighbours of $I_i^j$ in $S_{n,k}$; these are $I_i$, a node in $S_j$ (where, by definition, $S_j \notin \text{free}$) and $n - 3$ other neighbours. Consequently, from the $n - 3$ neighbours of $I_i^j$ different from $I_i$ and the neighbour in $S_j$, at least one, call it $(I_i^j)'$, is joined to a node in $S_i$ where $S_i \in \text{free}$. We choose such a node $(I_i^j)'$ in line 52, and in lines 53-55 we remove $\rho(I_i^j, I_i^j)$ from $S_i$-\text{paths} blocked, add the path $\rho(I_i^j, (I_i^j)') = (I_i^j, (I_i^j)')$ to $S_i$-\text{paths} and register that the nodes of $S_i$ are to be used as transit nodes to extend $\rho(I_i^j, (I_i^j)')$ to a path to $I_k$ and so can no longer be used as such for any other path. Note that $(I_i^j)'$ cannot lie on any path in $S_i$-\text{paths} because of our manipulation in lines 38-43. Also, the new path $\rho(I_i^j, (I_i^j)')$ does not contain a node of the form $(j', \ldots, i)$ for which $S_j' \in \text{free}$. We repeat the above for every path in $S_i$-\text{paths} blocked.

In lines 57-60, we extend all paths in $S_i$-\text{paths} (in the natural way) so that they reach $I_k$ and move the paths into \text{paths}. Thus, the procedure \text{Paths in some: target and blocked} achieves its aims.

Consider the situation where $S_i \in \text{blocked} \cap \text{some}$ and $I_i \notin T_i$ (recall, up until now we have assumed that $I_i \in T_i$). In order to deal with this situation we develop a new procedure \text{Paths in some: not target and blocked}$(S_i, T_i, I_i, \text{free, used paths})$. This procedure is very similar to \text{Paths in some: target and blocked} so we do not describe it in detail nor with pseudo-code, but only highlight any differences and comment on any amended analysis. To obtain \text{Paths in some: not target and blocked}, we omit lines 2-3 from \text{Paths in some: target and blocked} and amend line 4 so that temp is initialized as being empty. We omit lines 20-23 and amend line 24 to $I_i^j \in temp$. The analysis of \text{Paths in some: target and blocked} is identical to that of \text{Paths in some: not target and blocked}. Our only comment is that in the analysis corresponding to lines 5-15 of \text{Paths in some: target and blocked}, we still obtain that all neighbours $I_i^j$ of $I_i$ that are not target nodes and for which $S_j \notin \text{free}$ are chosen as temporary target nodes (the number of temporary target nodes chosen is $(n - 2) - |T_i|$ and the number of $S_i$'s, from $\{S_1, S_2, \ldots, S_n\} \setminus \{S_k, S_i\}$, that are not in $\text{free}$ is at most $(n - 1) - |T_i| - 1$). Furthermore, the analysis corresponding to line 52 of \text{Paths in some: target and blocked} still holds. Hence,
6.4. Building node-disjoint paths

Figure 6.2: Dealing with 'bad' target nodes.

Paths\_in\_some: not\_target\_and\_blocked achieves its aims.

6.4.4 Paths in $S_i \not\in blocked \cap some$

We are reduced to the situation where we have established some paths from target nodes in $S_k$ to $I_k$ (if there are any) and established some paths from target nodes in $S_i$ to $I_k$, where $S_i \in blocked \cap some$ (if such an $S_i$ exists). Thus, we have to deal with target nodes in other $S_j$'s for which $S_j \in some \setminus blocked$ (recall that there is at most one $S_i$ in $blocked \cap some$).

We deal with this situation with the procedure $Paths\_in\_some: not\_blocked(S_i, T_i, I_i, free, used, paths)$ (we have switched indices from $j$ to $i$ to make a comparison with $Paths\_in\_some: target\_and\_blocked$ easier). This procedure differs from $Paths\_in\_some: target\_and\_blocked$ sufficiently for it to be worthwhile detailing using pseudo-code. The line-numbering has been chosen so that $Paths\_in\_some: not\_blocked$ can more easily be compared with $Paths\_in\_some: target\_and\_blocked$.

As ever, we assume that $0 < |T_i| < n - 1$.

```
1  Paths\_in\_some: not\_blocked(S_i, T_i, I_i, free, used, paths)
4    temp := 0;
5  for each neighbour $I_j$ of $I_i$ in $S_i \setminus T_i$ do
6      if $S_j \not\in free$ and $|temp \cup (T_i \setminus \{I_i\})| < n - 2$ then
7        add $I_j$ to $temp$;
8  fi
9  od
```
6.4. Building node-disjoint paths

10 \text{temp.so.far} := \text{temp};
11 \text{for each neighbour } I_i^j \text{ of } I_i \text{ in } S_i \setminus (\text{temp.so.far} \cup T_i) \text{ do}
12 \quad \text{if } |\text{temp} \cup (T_i \setminus \{I_i\})| < n - 2 \text{ then}
13 \quad \quad \text{add } I_i^j \text{ to } \text{temp};
14 \quad \text{fi}
15 \quad \text{od}
16 \text{call Disjointpaths}(S_i, (\text{temp} \cup T_i) \setminus \{I_i\}, I_i, S_i\text{-paths});
17 S_i\text{-paths.blocked} := \emptyset;
17.1 \text{bad.target} := \epsilon;
17.2 \text{bad.terminal} := \epsilon;
18 \text{for every neighbour } I_i^j \text{ of } I_i \text{ in } S_i \text{ do}
19 \quad \text{case of}
24.1 I_i^j \in \text{temp} :
25 \quad \quad \text{remove the path } \rho(I_i^j, I_i) \text{ from } S_i\text{-paths};
26 \quad I_i^j \notin \text{temp} \cup T_i :
27 \quad \quad \text{replace the path } \rho(t, I_i) \text{ in } S_i\text{-paths}
28 \quad \quad \quad \text{upon which } I_i^j \text{ lies with the sub-path } \rho(t, I_i^j);
28.1 \quad \text{if } S_j \in \text{free} \text{ then}
28 \quad \quad \quad \quad \text{set } \text{escape}[\rho(t, I_i^j)] := S_j;
29 \quad \quad \quad \quad \text{remove } S_j \text{ from } \text{free} \text{ and add } S_j \text{ to } \text{used};
29.1 \quad \text{else}
29.2 \quad \quad \text{bad.target} := t;
29.3 \quad \quad \text{bad.terminal} := I_i^j;
29.4 \quad \text{fi}
30 I_i^j \in T_i \text{ and } S_j \in \text{free} :
31 \quad \text{replace the path } \rho(I_i^j, I_i) \text{ in } S_i\text{-paths}
32 \quad \quad \text{with the path } \rho(I_i^j, I_i^j) = (I_i^j);
32 \quad \quad \text{set } \text{escape}[\rho(I_i^j, I_i^j)] := S_j;
33 \quad \quad \text{remove } S_j \text{ from } \text{free} \text{ and add } S_j \text{ to } \text{used};
34 I_i^j \in T_i \text{ and } S_j \notin \text{free} :
35 \quad \text{remove the path } \rho(I_i^j, I_i) \text{ from } S_i\text{-paths} \text{ and add the}
path \( \rho(I_i^1, I_i^j) = (I_i^j) \) to \( S_i\_paths\_blocked; \)

```latex
36 \text{ esac;}
37 \text{ od}
38 \text{ while some path } \rho(t, s) \text{ in } S_i\_paths \text{ contains a node whose first component } j, \text{ say, is such that } S_j \in \text{ free do}
39 \text{ replace the path } \rho(t, s) \text{ in } S_i\_paths \text{ with its sub-path } \rho(t, x) \text{ where } x \text{ is such that its first component } j, \text{ say, is such that } S_j \in \text{ free and where if } y \neq x \text{ is any other node on } \rho(t, x) \text{ then its first component } j', \text{ say, is such that } S_{j'} \notin \text{ free;}
39.1 \text{ if } t \neq \text{ bad.target or } s \neq \text{ bad.terminal then}
40 \text{ remove escape[ } \rho(t, s) \text{ ] from } \text{ used and add escape[ } \rho(t, s) \text{ ] to } \text{ free;}
40.1 \text{ fi}
41 \text{ set } escape[ \rho(t, x) ] := S_j;
42 \text{ add } S_j \text{ to } \text{ used and remove } S_j \text{ from } \text{ free;}
43 \text{ od}
44 \text{ for every path } \rho(I_i^1, I_i^j) \in S_i\_paths\_blocked \text{ do}
45 \text{ if } S_j \in \text{ free then}
46 \text{ remove } \rho(I_i^1, I_i^j) \text{ from } S_i\_paths\_blocked
47 \text{ and add } \rho(I_i^2, I_i^j) \text{ to } S_i\_paths;
48 \text{ set } escape[ \rho(I_i^2, I_i^j) ] := S_j;
49 \text{ remove } S_j \text{ from } \text{ free and add } S_j \text{ to } \text{ used;}
50 \text{ fi}
50.1 \text{ od}
50.2 \text{ if } I_i \in T_i \text{ then}
50.3 \text{ else}
50.4 \text{ if } \text{ bad.target } \neq \epsilon \text{ and } \text{ bad.terminal } \neq \epsilon \text{ then}
50.5 \text{ if } \rho(\text{bad.target, bad.terminal}) \in S_i\_paths \text{ then}
```

This snippet of text describes a process for building node-disjoint paths, involving the manipulation of paths and sets. It includes operations like replacing paths, checking conditions, and updating sets and escape paths. The conditions involve checking for paths containing nodes and manipulating these paths based on certain conditions.
remove $\rho(bad\_target, bad\_terminal)$ from $S_i$-paths;

\textbf{else}

remove the path $\rho(bad\_target, s)$ from $S_i$-paths;

remove $escape[\rho(bad\_target, s)]$ from used and add $escape[\rho(bad\_target, s)]$ to free;

\textbf{fi}

\textbf{extend} $\rho(bad\_target, bad\_terminal)$ through $I_i$ to $I_k$

and call this path $\rho(bad\_target, I_k)$;

add $\rho(bad\_target, I_k)$ to paths;

\textbf{else}

\textbf{if} $S_i$-paths\_blocked $\neq \emptyset$ then

choose some $\rho(I_i^j, I_k^j) \in S_i$-paths\_blocked,

remove it from $S_i$-paths\_blocked and add the path $\rho(I_i^j, I_k^j) = (I_i, I_i^j, I_k)$ (resp. $\rho(I_i^j, I_k^j) = (I_i, I_i^j, I_k)$)

to paths if $i \neq 1$ (resp. $i = 1$);

\textbf{else}

remove some path $\rho(t, s)$ from $S_i$-paths;

remove $escape[\rho(t, s)]$ from used and

add $escape[\rho(t, s)]$ to free;

\textbf{define} the path $\rho(t, I_k)$ by extending the

original path $\rho(t, I_i^j)$ of which $\rho(t, s)$ is a

sub-path to $I_i$ and then on to $I_k$;

add $\rho(t, I_k)$ to paths;

\textbf{fi}

\textbf{for every} path $\rho(I_i^j, I_k^j) \in S_i$-paths\_blocked do

choose a neighbour $(I_i^j)^i$ of $I_i^j$ in $S_i$ for which $S_i \in free$;

remove $\rho(I_i^j, I_k^j)$ from $S_i$-paths\_blocked and

add $\rho(I_i^j, (I_i^j)^i) = (I_i, (I_i^j)^i)$ to $S_i$-paths;

set $escape[\rho(I_i^j, (I_i^j)^i)] := S_i$;
remove \( S_i \) from \( \text{free} \) and add \( S_i \) to \( \text{used} \);

for every path \( \rho(t,s) \) in \( S_i \)-paths do

extend \( \rho(t,s) \) to a path \( \rho(t,I_k) \) through the nodes of

\( S_j = \text{escape}[\rho(t,s)] \) to \( I_j \) and then on to \( I_k \);

remove \( \rho(t,s) \) from \( S_i \)-paths and add \( \rho(t,I_k) \) to \( \text{paths} \);

od

We explain below what \texttt{Paths.in.some:not.blocked} does, and prove that what the procedure claims to do is actually possible and that it achieves its aims. It has some similarities with \texttt{Paths.in.some:target.and.blocked} and so we are brief with some of the analysis below when this analysis is identical to before with \texttt{Paths.in.some:target.and.blocked}. We assume that (after possible calls to \texttt{Paths.in.S_k} and \texttt{Paths.in.some:target.and.blocked}), it is the case that \(|\text{free}| = (n - 1) - |T_i|\), if there was no call to \texttt{Paths.in.some:target.and.blocked}, and \(|\text{free}| \geq (n - 2) - (|T_k| - 1) - |T_{t_0}| = (n - 1) - |T_k| - |T_{t_0}|\), where \( S_{t_0} \) is the focus of the call to \texttt{Paths.in.some:target.and.blocked}. Thus, regardless, \(|\text{free}|\) is \( n - 1 \) minus the total number of target nodes in the \( S_j \)'s 'dealt with' so far.

In lines 4-15, neighbours of \( I_i \) are set as temporary target nodes, making sure that neighbours \( I_i^j \) for which \( S_j \not\in \text{free} \) are chosen before neighbours \( I_i^j \) for which \( S_j \in \text{free} \). The process stops when: \(|T_i| - 1\) plus the number of temporary target nodes is exactly \( n - 2 \), if \( I_i \in T_i \); or when \(|T_i|\) plus the number of temporary target nodes is exactly \( n - 2 \), if \( I_i \not\in T_i \). We claim that: if \( I_i \in T_i \) then all neighbours \( I_i^j \) of \( I_i \) in \( S_i \) that are not target nodes and for which \( S_j \not\in \text{free} \) are chosen as temporary target nodes; and that if \( I_i \not\in T_i \) then all neighbours \( I_i^j \) of \( I_i \) in \( S_i \) that are not target nodes and for which \( S_j \not\in \text{free} \) are chosen as temporary target nodes except possibly for at most one such neighbour. We now verify this claim.

Suppose that the call to \texttt{Paths.in.some:target.and.blocked} was made. Let us count the number of \( S_i \)'s, from \( \{S_1, S_2, \ldots, S_n\} \setminus \{S_k, S_{t_0}, S_i\} \), that are not in \( \text{free} \). The reason any \( S_i \not\in \text{free} \) is that exactly one of the following holds: \( S_i \in \text{blocked} \); \( S_i \in \text{some} \setminus \text{blocked} \); and \( S_i \in \text{used} \setminus \text{blocked} \). We can associate a target node with each such \( S_i \): if \( S_i \in \text{blocked} \) then choose the target node in \( T_k \) on whose
path in \textit{paths} the (blocking) node $I_k^i$ lies; if $S_i \in \textit{some \setminus blocked}$ then choose any target node in $S_i$; and if $S_i \in \textit{used \setminus blocked}$ then choose the unique target node in $S_i$ on whose path in \textit{paths} the nodes of $S_i$ are used as transit nodes. Note that distinct $S_i$'s are associated with distinct target nodes. Thus, the number of $S_i$'s, from $\{S_1, S_2, \ldots, S_n\} \setminus \{S_k, S_{i_0}, S_i\}$, that are not in \textit{free} is at most the number of target nodes that potentially can be associated with such an $S_i$, and this is $(|T_k| - 1) + |T_{i_0}| + \sum_{j \neq k, i_0} |T_j| = (n - 2) - |T_i|$ (note that $I_k^{i_0} \in T_k$ cannot be so associated). Thus, irrespective of whether $I_i$ is in $T_i$ or not, all neighbours $I_j^i$ of $I_i$ in $S_i$ that are not target nodes and for which $S_j \notin \textit{free}$ are chosen as temporary target nodes.

Suppose that the call to \texttt{Paths.in\_some:target\_and\_blocked} was not made. Let us count the number of $S_i$'s, from $\{S_1, S_2, \ldots, S_n\} \setminus \{S_k, S_i\}$, that are not in \textit{free}. As above, the number of such $S_i$'s is at most $|T_k| + \sum_{j \neq k, i} |T_j| = (n - 1) - |T_i|$. Thus, if $I_i \in T_i$ then all neighbours $I_j^i$ of $I_i$ in $S_i$ that are not target nodes and for which $S_j \notin \textit{free}$ are chosen as temporary target nodes; however, if $I_i \notin T_i$ then there may be at most one such neighbour that is not chosen as a temporary target node. Hence, our claim holds.

In line 16, we recursively find node-disjoint paths in $S_i$ from every target node of $T_i \setminus \{I_i\}$ and every temporary target node to the node $I_i$, as we did before. In line 17, $S_i$-\texttt{paths.blocked} is initialized as an empty set of paths, and in lines 17.1-17.2, the nodes \texttt{bad\_target} and \texttt{bad\_terminal} are set as $\epsilon$, i.e., 'nil'.

In lines 18-37, we amend each path in $S_i$-\texttt{paths} by working through the neighbours $I_j^i$ of $I_i$ in $S_i$ in turn as follows. If $I_j^i \in \textit{temp}$ then we remove the path $\rho(I_j^i, I_i)$ from $S_i$-\texttt{paths}. If $I_j^i \notin \textit{temp} \cup T_i$ then we truncate the path $\rho(t, I_i)$ containing $I_j^i$ by removing the final edge. We also register that the nodes of $S_j$ are to be used as transit nodes for this truncated path, but only if the path in question is such that $S_j \in \textit{free}$; otherwise, we set \texttt{bad\_target} $= t$ and \texttt{bad\_terminal} $= I_j^i$ (from above, there is at most one such path). If $I_j^i \in T_i$ then we truncate the path $\rho(I_j^i, I_i)$ containing $I_j^i$ by removing the final edge. If $S_j \in \textit{free}$ then we register that the nodes of $S_j$ are to be used as transit nodes for this truncated path; otherwise, we move the path $\rho(I_j^i, I_i)$ from $S_i$-\texttt{paths} to $S_i$-\texttt{paths.blocked}. 
6.4. Building node-disjoint paths

In line 38-43, we amend the paths of $S_i\text{-paths}$. Suppose that some path $\rho(t, s)$ in $S_i\text{-paths}$, where $t \neq bad\_target$ or $s \neq bad\_terminal$, is such that there is some node $x$ of the form $(j, \ldots, i)$ lying upon it so that $S_j \in free$. We can replace the path $\rho(t, s)$ in $S_i\text{-paths}$ with the sub-path $\rho(t, x)$, so long as we release the set of transit nodes $\text{escape}[\rho(t, s)]$ (for possible future use) and register that the new set of transit nodes $S_j$ is not to be used as a set of transit nodes for any other path. If the path $\rho(bad\_target, bad\_terminal)$ is such that there is some node $x$ of the form $(j, \ldots, i)$ lying upon it so that $S_j \notin free$ then we can simply replace it in $S_i\text{-paths}$ with the path $\rho(bad\_target, x)$ and register that the new set of transit nodes $S_j$ is not to be used as a set of transit nodes for any other path (for if $bad\_terminal = I_i^j$ then $S_i \notin free$). By iterating this process, we get to the situation where no path in $S_i\text{-paths}$ contains a node of the form $(j, \ldots, i)$ so that $S_j \notin free$.

In lines 44-50, we deal with some of the paths in $S_i\text{-paths}\_blocked$, each of which is of the form $\rho(I_i^j, I_i^l)$, as we did before.

In lines 50.1-50.n, we ensure that exactly one path from a chosen target node in $S_i$ to $I_k$ will pass through the node $I_i$ (after construction, this path is placed in $paths$). If $I_i \in T_i$ then our chosen target node is $I_i$; recall from earlier that when $I_i \in T_i$, there is no 'bad' path $\rho(bad\_target, bad\_terminal)$ and so all other paths in $S_i\text{-paths}$ can be extended through transit nodes (as is done in lines 57-60). Otherwise, if the 'bad' path $\rho(bad\_target, bad\_terminal)$ exists and still resides in $S_i\text{-paths}$ then we extend this path through $I_i$ to $I_k$; alternatively, if there is a path of the form $\rho(bad\_target, s)$ in $S_i\text{-paths}$ (where $s \neq bad\_terminal$) then we release the corresponding set of transit nodes for possible future use before replacing $\rho(bad\_target, s)$ with the extension of the original path $\rho(bad\_target, bad\_terminal)$ through $I_i$ to $I_k$. Suppose that $I_i \notin T_i$ and that no 'bad' path has initially been registered. If $S_i\text{-paths}\_blocked$ is non-empty then some path from $S_i\text{-paths}\_blocked$ is chosen and extended through $I_i$ to $I_k$. Alternatively, if $S_i\text{-paths}\_blocked$ is empty then some path $\rho(t, s)$ from $S_i\text{-paths}$ is chosen and replaced with the original path $\rho(t, I_i^l)$ of which $\rho(t, s)$ is a sub-path; the path $\rho(t, I_i^l)$ is extended through $I_i$ to $I_k$ and the set of transit nodes corresponding to $\rho(t, s)$ is released for possible future use. Irrespective of which path is chosen to go through $I_i$, note that the corresponding
target node is not associated with any set of transit nodes.

The subsequent execution of the algorithm is as before; however, we must verify that a node \((I_i^j)^j\) can be chosen as in line 52. We claim that there exists a neighbour \((I_i^j)^j\) of \(I_i^j\) in \(S_i \setminus \{I_i\}\) such that \(S_i \in \text{free}\).

Suppose that the call to \texttt{Paths.in\_some\_target\_and\_blocked} was made. Let us count the number of \(S_i\)'s, from \(\{S_1, S_2, \ldots, S_n\} \setminus \{S_k, S_{i_0}, S_i\}\), for which \(S_i \notin \text{free}\). As we have seen already, any such \(S_i\) can be associated with a target node and all these associated target nodes are distinct. The maximum number of target nodes eligible to be associated with such an \(S_i\) is \((n - 1) - \alpha - 2\), where \(\alpha\) is the number of paths currently in \(S_i\_paths\_blocked\) (remember, from the last line of the preceding paragraph, there is a target node of \(S_i\), but not one of the \(\alpha\) target nodes, not associated with any set of transit nodes; also, \(I_i^0 \in T_k\) is not so associated). Hence, the number of \(S_i\)'s, from \(\{S_1, S_2, \ldots, S_n\} \setminus \{S_k, S_{i_0}, S_i\}\), for which \(S_i \in \text{free}\) is at least \((n - 3) - ((n - 1) - \alpha - 2) = \alpha \geq 1\). Consider the neighbours of \(I_i^j\) in \(S_{n,k}\); these are \(I_i\), a node in \(S_j\) (where, by definition, \(S_j \notin \text{free}\)) and \(n - 3\) other neighbours.

Consequently, from the \(n - 3\) neighbours of \(I_i^j\) different from \(I_i\) and the neighbour in \(S_j\), at least one, call it \((I_i^j)^j\), is joined to a node in \(S_i\) where \(S_i \in \text{free}\).

Suppose that the call to \texttt{Paths.in\_some\_target\_and\_blocked} was not made. Let us count the number of \(S_i\)'s, from \(\{S_1, S_2, \ldots, S_n\} \setminus \{S_k, S_i\}\), for which \(S_i \notin \text{free}\). The maximum number of target nodes eligible to be associated with such an \(S_i\) is \((n - 1) - \alpha - 1\), where \(\alpha\) is the number of paths currently in \(S_i\_paths\_blocked\) (again, there is a target node of \(S_i\), but not one of the \(\alpha\) target nodes, not associated with any set of transit nodes). Hence, the number of \(S_i\)'s, from \(\{S_1, S_2, \ldots, S_n\} \setminus \{S_k, S_i\}\), for which \(S_i \in \text{free}\) is at least \((n - 2) - ((n - 1) - \alpha - 1) = \alpha \geq 1\). As above, the required node \((I_i^j)^j\) exists and our claim holds.

It can easily be verified that if we repeatedly apply the procedure \texttt{Paths.in\_some\_not\_blocked} then the analysis as presented above still holds true (essentially because every time we apply \texttt{Paths.in\_some\_not\_blocked}, one of the target nodes in the \(S_i\) in question is always on a path through \(I_i\) to \(I_k\) and thus does not use any set of transit nodes). Consequently, the procedure \texttt{Paths.in\_some\_not\_blocked} achieves its aims, as does our main algorithm \texttt{Disjoint\_paths}. 

### 6.4. Building node-disjoint paths

139
6.5 Path lengths and complexity

Having proved that our algorithm Disjoint_paths finds a collection of node-disjoint paths in $S_{n,k}$ from $n - 1$ target nodes to a source node, we now turn to the lengths of the paths produced by the algorithm and the time complexity of the algorithm.

We derive below an upper bound on the length of any path constructed by Disjoint_paths; in the first instance, this upper bound is in the form of a recurrence relation. Let $b_k$ be an upper bound on the length of any path produced by the algorithm Disjoint_paths applied in $S_{n,k}$, irrespective of $n$ (at the moment, we have not shown that such an upper bound exists; however, we show, using induction, that it does and derive an estimate of it). By Theorem 6.3.1, $b_2 = 5$. In order to derive the recurrence relation, we consider each of the procedures Paths_in_Sn, Paths_in_some:target_and_blocked, Paths_in_some:not_target_and_blocked and Paths_in_some:not_blocked in turn (when called from within Disjoint_paths applied in $S_{n,k}$, where $k$ is at least 3). As our induction hypothesis, we assume that $b_{k-1}$ exists.

The following lemma proves useful.

**Lemma 6.5.1** Let $(j, x_2, \ldots, x_{k-1}, i)$ and $(y_1, y_2, \ldots, y_{k-1}, i)$ be nodes of $S_i$ in $S_{n,k}$, for some $i, j \in \{1, 2, \ldots, n\} \setminus \{k\}$, with $i \neq j$, and let $\rho((j, x_2, \ldots, x_{k-1}, i), (y_1, y_2, \ldots, y_{k-1}, i))$ be a path in $S_i$ of length $t$. Also, let $(z_1, z_2, \ldots, z_{k-1}, j)$ be the node of $S_j$ such that for every $l = 1, 2, \ldots, k - 1$, if $y_l \neq j$ then $z_l = y_l$, and if $y_l = j$ then $z_l = i$.

(a) There is a path $\rho((i, x_2, \ldots, x_{k-1}, j), (z_1, z_2, \ldots, z_{k-1}, j))$ in $S_j$ of length $t$.

(b) If, further, $(y_1, y_2, \ldots, y_{k-1}, i) = I_i$ then there is a path from $(z_1, z_2, \ldots, z_{k-1}, j)$ to $I_j$ in $S_j$ of length at most 3.

**Proof:** (a) This follows from a simple induction on the length of the path $\rho((j, x_2, \ldots, x_{k-1}, i), (z_1, z_2, \ldots, z_{k-1}, i))$.

(b) There are a number of cases to consider. Denote $(z_1, z_2, \ldots, z_{k-1}, j)$ by $Z$.

Case (i): Suppose that $I_i = (k, 2, \ldots, i - 1, 1, i + 1, \ldots, k - 1, i)$, where $i \in \{2, 3, \ldots, k - 1\}$, and that $I_j = (k, 2, \ldots, j - 1, 1, j + 1, \ldots, k - 1, j)$, where $j \in \{2, 3, \ldots, k -
Thus, $Z = (k, 2, \ldots, i - 1, 1, i + 1, \ldots, j - 1, i, j + 1, \ldots, k - 1, j)$ and there is a path from $Z$ to $I_j$ of length at most 3.

Case (ii): Suppose that $I_i = (k, 2, \ldots, i - 1, 1, i + 1, \ldots, k - 1, i)$, where $i \in \{2, 3, \ldots, k - 1\}$, and that $I_j = (k, 2, \ldots, k - 1, j)$, where $j \in \{k + 1, k + 2, \ldots, n\}$.

Thus, $Z = (k, 2, \ldots, i - 1, 1, i + 1, \ldots, k - 1, j)$ and there is a path from $Z$ to $I_j$ of length at most 3.

Case (iii): Suppose that $I_i = (k, 2, \ldots, i - 1, 1, i + 1, \ldots, k - 1, i)$, where $i \in \{2, 3, \ldots, k - 1\}$, and that $j = 1$ with $I_1 = (k, 2, \ldots, k - 1, 1)$.

Thus, $Z = (k, 2, \ldots, k - 1, 1)$ and $Z = I_1$.

Case (iv): Suppose that $I_i = (k, 2, \ldots, k - 1, i)$, where $i \in \{k + 1, k + 2, \ldots, n\}$, and that $I_j = (k, 2, \ldots, j - 1, 1, j + 1, \ldots, k - 1, j)$, where $j \in \{2, 3, \ldots, k - 1\}$.

Thus, $Z = (k, 2, \ldots, j - 1, i, j + 1, \ldots, k - 1, j)$ and there is a path from $Z$ to $I_j$ of length at most 3.

Case (v): Suppose that $I_i = (k, 2, \ldots, k - 1, i)$, where $i \in \{k + 1, k + 2, \ldots, n\}$, and that $I_j = (k, 2, \ldots, k - 1, j)$, where $j \in \{k + 1, k + 2, \ldots, n\} \setminus \{i\}$.

Thus, $Z = (k, 2, \ldots, k - 1, j)$ and $Z = I_j$.

Case (vi): Suppose that $I_i = (k, 2, \ldots, k - 1, i)$, where $i \in \{k + 1, k + 2, \ldots, n\}$, and that $j = 1$ with $I_1 = (k, 2, \ldots, k - 1, 1)$.

Thus, $Z = (k, 2, \ldots, k - 1, 1)$ and $Z = I_1$.

Case (vii): Suppose that $i = 1$ with $I_1 = (k, 2, \ldots, k - 1, 1)$ and that $I_j = (k, 2, \ldots, j - 1, 1, j + 1, \ldots, k - 1, j)$, for some $j \in \{2, 3, \ldots, k - 1\}$.

Thus, $Z = (k, 2, \ldots, j - 1, 1, j + 1, \ldots, k - 1, j)$ and $Z = I_j$.

Case (viii): Suppose that $i = 1$ with $I_1 = (k, 2, \ldots, k - 1, 1)$ and that $I_j = (k, 2, \ldots, k - 1, j)$, for some $j \in \{k + 1, k + 2, \ldots, n\}$.

Thus, $Z = (k, 2, \ldots, k - 1, j)$ with $Z = I_j$.

The result follows. □

Trivially, every path produced by Paths.in.\_S_k has length $b_{k-1}$. Consider the paths constructed by Paths.in.\_some: target_and_blocked. Each path begins as
6.5. Path lengths and complexity

a path of length at most $b_k-1$ produced by the recursive call to \texttt{Disjoint.paths}. Some paths are essentially constructed in lines 18-50 (except that they need to be extended through the appropriate $S_j$ to $I_j$ and then on to $I_k$); others are essentially constructed in lines 51-60. Consider a path $\rho(t, I_k)$ constructed according to lines 18-50. In general, this path: starts out as a path $\rho(t, I_i)$ of length at most $b_k-1$; is progressively shortened so that some sub-path from some node $X$ of $\rho(t, I_i')$ to $I_i'$ is removed; and the sub-path $\rho(t, X)$ is extended through some $S_i$ to $I_i$ and then on to $I_k$. By Lemma 6.5.1, the resulting (sub-)path from $t$ to $I_i$ has length at most $b_k-1 + 4$, and so the resulting path $\rho(t, I_k)$ has length at most $b_k-1 + 6$.

Consider a path $\rho(t, I_k)$ constructed according to lines 51-60. Again by Lemma 6.5.1, this path has length at most 9. The same path-length analysis holds for both \texttt{Paths.in.some:not.target.and.blocked} and \texttt{Paths.in.some:not.blocked}. Thus, we have that $b_k$ exists and $b_k < b_k-1 + 6$. Thus, by induction and as $b_2 = 5$, we have that $b_k \leq 6k - 7$.

As regards the time complexity of our algorithm, consider the execution of \texttt{Disjoint.paths} on $S_{n,k}$, with the set of target nodes $T$ and with the source node $I$. This execution results in a tree $\tau$ describing the procedure calls, with every node of the tree $\tau$ corresponding to a call of the procedure \texttt{Disjoint.paths}, \texttt{Paths.in.S_k}, \texttt{Paths.in.some:target.and.blocked}, \texttt{Paths.in.some:not.target.and.blocked} or \texttt{Paths.in.some:not.blocked}, as follows: a node corresponding to some procedure $P$ has a child corresponding to some procedure $Q$ if a call is made to procedure $Q$ from within the call to procedure $P$. The structure of the tree $\tau$ can be visualized as in Fig.6.3, where a node is labelled $D$ (a $D$-node) if it corresponds to a call of the procedure \texttt{Disjoint.paths} and $P$ (a $P$-node) if it corresponds to a call of one of the other 4 procedures. Note that it may be the case that a $D$-node has only 1 child; however, every $P$-node has exactly one child.

We can associate with each $D$-node of $\tau$ a pair of integers $(k, t)$ if the particular call involves $S_{n,k}$, the target nodes $T$ and the source node $I$, and if there are $t$ target nodes of $T$ not adjacent to the source node $I$. Note that the pair of integers associated with the root of $\tau$ is $(k, t)$, for some $t \leq n$. If a $D$-node $u$ has an associated pair $(m, t)$ and $d$ children then a simple consideration of the procedure calls detailed...
in the algorithm, \texttt{Disjoint paths} yields that the pair associated with the unique 
\( D \)-child of the \( i \)-th \( P \)-child of \( u \) must be of the form \((m - 1, t_i)\) and we must have 
that \( t_1 + t_2 + \ldots + t_d \leq t \).

Remove all \( P \)-nodes from \( \tau \) by inserting an edge joining the parent and the child 
of any \( P \)-node; denote the resulting tree by \( \tau' \). We claim that \( \tau' \) has at most \((k - 2)t^2\) 
edges, where the pair of integers associated with the root is \((k, t)\), for some \( k \geq 2 \), 
and we prove this claim by induction on \( k \) (the base case, when \( k = 2 \), trivially holds). Suppose that the root has \( d \) children and that the pair of integers associated 
with the \( i \)-th child is \((k - 1, t_i)\); so, in particular, \( t_1 + t_2 + \ldots + t_d \leq t \). By the induction 
hypothesis, the sub-tree rooted at the \( i \)-th child of the root has at most \((k - 3)t_i^2\) 
edges. Thus, the number of edges in \( \tau' \) is at most \((k - 3)(t_1^2 + t_2^2 + \ldots + t_d^2) + d \) edges, 
which in turn is at most \((k - 2)(t_1 + t_2 + \ldots + t_d)^2 \leq (k - 2)t^2\). Hence, our claim 
holds.

The upshot is that in any execution of \texttt{Disjoint paths} on \( S_{n,k} \) with a set of 
\( \text{target nodes} \) \( T \) of size \( n - 1 \), there are at most \( 2(k - 2)(n - 1)^2 \) procedure calls. Given 
that \( b_k \leq 6k - 7 \), it is trivial to see that apart from a call to another procedure, 
all procedures take \( O(k^2n^2) \) time, as does the procedure \texttt{Disjoint paths when } \( k=2 \) 
(by Theorem 6.3.1). Hence, \texttt{Disjoint paths} on \( S_{n,k} \) with a set of \( \text{target nodes} \) \( T \) 
of size \( n - 1 \) has time complexity \( O(k^3n^4) \).
6.6 Conclusions

In this chapter, we have derived a polynomial-time algorithm to find node-disjoint paths from each of $n - 1$ distinct target nodes in $S_{n,k}$ to a source node (different from any target node). The length of any path constructed is at most $6k - 7$. This should be compared with the diameter of $S_{n,k}$ which is at most $2k - 1$ (see the Introduction for an exact formula for the diameter of $S_{n,k}$).

Of course, we can apply our algorithm to $S_{n-1,n}$, i.e., the $n$-star. What results is an algorithm of time complexity $O(n^7)$ that finds node-disjoint paths, each of length at most $6n - 13$. As might be expected, the algorithm from [34], designed specifically for $n$-stars, is better in that it has time complexity $O(n^2)$ and results in node-disjoint paths each of length at most $3n + 9$. Similarly, we can apply our algorithm to produce a $(u, v)$-container, for distinct nodes $u$ and $v$ of $S_{n,k}$. Again, as expected, the resulting container is much worse than that produced by the (polynomial-time) algorithm in [115] (specifically designed for the purpose) where one of wide-diameter at most $2k + 1$ is produced. Nevertheless, our algorithm gives a polynomial-time alternative for constructing node-disjoint paths in $n$-stars and containers in $S_{n,k}$. 
Chapter 7

Conclusion and future work

There are many studies on different aspects of interconnection networks for parallel and distributed computing; for example, the topological properties of different interconnection networks, routing and communication algorithms designed for interconnection networks, and fault-tolerant properties of interconnection networks. In this thesis we considered several properties for $k$-ary $n$-cubes and $(n,k)$-star graphs, and we proposed a new interconnection network, the augmented $k$-ary $n$-cube. In detail, we obtained the following results:

1. Let $k \geq 4$ be even and let $n \geq 2$. Consider a faulty $k$-ary $n$-cube $Q^k_n$ in which the number of node faults $f_v$ and the number of link faults $f_e$ are such that $f_v + f_e \leq 2n - 2$. We prove that given any two healthy nodes $s$ and $e$ of $Q^k_n$, there is a path from $s$ to $e$ of length at least $k^n - 2f_v - 1$ (resp. $k^n - 2f_v - 2$) if the nodes $s$ and $e$ have different (resp. the same) parities (the parity of a node in $Q^k_n$ is the sum modulo 2 of the elements in the $n$-tuple over $\{0,1,\ldots,k-1\}$ representing the node). Our result is optimal in the sense that there are pairs of nodes and fault configurations for which these bounds cannot be improved, and it answers questions recently posed by Yang, Tan and Hsu, and by Fu. Furthermore, we extend known results, obtained by Kim and Park, for the case when $n = 2$.

2. We give precise solutions to problems posed by Wang, An, Pan, Wang and Qu and by Hsieh, Lin and Huang. In particular, we show that $Q^k_n$ is bi-
panconnected and edge-bipancyclic, when \( k \geq 3 \) and \( n \geq 2 \), and we also show that when \( k \) is odd, \( Q_n^k \) is \( m \)-panconnected, for \( m = \frac{n(k-1)+2k-6}{2} \), and \( (k-1) \)-pancyclic (these bounds are optimal). We introduce a path-shortening technique, called progressive shortening, and strengthen existing results, showing that when paths are formed using progressive shortening then these paths can be efficiently constructed and used to solve a problem relating to the distributed simulation of linear arrays and cycles in a parallel machine whose interconnection network is \( Q_n^k \), even in the presence of a faulty processor.

3. We define an interconnection network \( AQ_{n,k} \) which we call the augmented \( k \)-ary \( n \)-cube by extending a \( k \)-ary \( n \)-cube in a manner analogous to the existing extension of an \( n \)-dimensional hypercube to an \( n \)-dimensional augmented cube. We prove that the augmented \( k \)-ary \( n \)-cube \( AQ_{n,k} \) has a number of attractive properties (in the context of parallel computing). For example, we show that the augmented \( k \)-ary \( n \)-cube \( AQ_{n,k} \) is a Cayley graph (and so is vertex-symmetric); has connectivity \( 4n-2 \), and is such that we can build a set of \( 4n-2 \) mutually disjoint paths joining any two distinct vertices so that the path of maximal length has length at most \( \max\{(n-1)k-(n-2),k+7\} \); has diameter \( \lfloor \frac{k}{3} \rfloor + \lceil \frac{k-1}{3} \rceil \), when \( n = 2 \); and has diameter at most \( \frac{k}{4}(n+1) \), for \( n \geq 3 \) and \( k \) even, and at most \( \frac{k}{4}(n+1) + \frac{n-1}{3} \), for \( n \geq 3 \) and \( k \) odd.

4. We present an algorithm which given a source node and a set of \( n-1 \) target nodes in the \((n,k)\)-star graph \( S_{n,k} \), where all nodes are distinct, builds a collection of \( n-1 \) node-disjoint paths, one from each target node to the source. The collection of paths output from the algorithm is such that each path has length at most \( 6k-7 \), and the algorithm has time complexity \( O(k^3n^4) \).

Our research plan in the near future may focus on the following topics:

- Tolerating faults under conditional fault assumptions: to classify the fault-tolerance of interconnections networks used within parallel computing with respect to path- and cycle-based properties and under conditional fault assumptions.
7. Conclusion and future work

- Tolerating faults in OTIS networks: to investigate further the topological and algorithmic properties of general OTIS networks (Optical Transpose Interconnect System network [121,173]), both in the absence and presence of faults.

- The distributed construction of embedded structures: to investigate further the distributed construction of embedded structures within faulty interconnection networks.
Bibliography


[102] Cheng-Nan Lai, Gen-Huey Chen, and Dyi-Rong Duh. Constructing one-to-
2002.


[105] S. Latifi, S.Q. Zheng, and N. Bagherzadeh. Optimal ring embedding in hyper-
gest of Papers. Twenty-Second International Symposium on, pages 178–184,
Boston, MA, USA, 1992.


[108] F. Thomson Leighton. Introduction to parallel algorithms and architectures:
array, trees, hypercubes. Morgan Kaufmann Publishers Inc., San Francisco,
CA, USA, 1992.

[109] M. Lewinter and W. Widulski. Hyper-hamilton laceable and caterpillar-

[110] T.K. Li et al. Bipanconnectivity and edge-fault-tolerant bipancyclicity of hyper-


[113] TsengKuei Li, Jimmy J.M. Tan, and LihHsing Hsu. Hyper hamiltonian lace-

[114] Sheng-Chyang Liaw, Gerald J. Chang, Feng Cao, and D. Frank Hsu. Fault-


[118] Meijie Ma, Guizhen Liu, and JunMing Xu. Panconnectivity and edge-fault-


[124] Menn and Somani. An efficient sorting algorithm for the star graph intercon-


Appendix A

Source code: verify the base case of Theorem 3.3.3

'In the front, the author would like to give the following declaration: The program is solely coded by Mr. Yonghong Xiang at Department of Computer Science Department, Durham University while he was doing his Ph.D study from 2005--2008 in Durham.

'The purpose of the program is: to check the base case in proving that one can find a longest possible path in k-ary n-cube. That is, for 2-ary 4-cube, the program proves the result holds in the following aspects:

'1) If there are two faulty nodes in 2-ary 4-cube, there exist paths of length at least 10 or 11 between any two given healthy nodes.

'2) If there are one faulty node in 2-ary 4-cube, there exist paths of length at least 12 or 13 between any two given healthy nodes.

'3) If there are one faulty node and one faulty edge in 2-ary 4-cube, there exist paths of length at least 12 or 13 between any two given healthy nodes.

'4) If there are two faulty edges in 2-ary 4-cube, there exist paths of length at least 14 or 15 between any two given healthy nodes.
'The results of the program are saved as a .txt file in C:\, which
'show from one specific node, there exist path of specific length to
'all other possible nodes.

'In the program, we didn't check all possible cases, but only several
'cases. That is because by the symmetric properties of the 2-ary
'4-cube, all other cases can be obtained from
'the existing cases by some mapping function.

'In the following codes, the paths are not printed out. But the
'author has put all the necessary codes there.
'Once remove the corresponding comment symbol, one will get them.
'One might further improve the code by combining sub procedure
'FindPath and FindPathEdgeFaulty with one sub procedure. But, the
'current version works.

'Environment requirements: Windows XP, Visual Basic 6.0.

'If you need any more information, please contact the author:
'Mr Yonghong Xiang
'e-mail: yh_xiang@hotmail.com
'Last update: 18-11-2008

Private Type nodeDef
    IndexN As Integer
    Direction As Integer
    BDirection(4) As Boolean ' to indicate whether this edge is fault.
    'One faulty edge means there are two nodes should set some boolean
    'value false.
End Type
A. Source code: verify the base case of Theorem 3.3.3

Dim Node(15) As nodeDef
Const Rt = 1, Dn = 2, Lf = 3, Up = 4

Dim Path(571200, 15) As Integer
' remember all possible paths from one fixed node to any other node.
Dim PathEnds(15, 15) As Integer
' to remember the end nodes of each start node,
' so that we can conclude that the theorem is true.
' set its initial value as -1.
Dim totalEnds As Integer
' remember the number of end nodes from one fixed node with longest
' length.

Dim PaNu As Integer
Dim Addr As String

Private Sub InitNode() ' to initialize all nodes, ready for use.
Dim j As Integer, i As Integer
For j = 0 To 15
    Node(j).IndexN = j
    Node(j).Direction = 0
    For i = 0 To 4
        Node(j).BDirection(i) = True
    Next i
Next j
End Sub

Private Sub FindPath(startNode As nodeDef, PLen As Integer)
Dim str(15) As Integer ' remember the index of the current path
Dim ss As String ' used to print information to file
Dim i As Integer, j As Integer
A. Source code: verify the base case of Theorem 3.3.3

```vbnet
Dim lenP As Integer 'remember the current considering path's length
Dim Dict As Integer 'remember the direction of the current considering node
Dim PathNum As Integer 'remember the number of paths
Dim saveStartNode As Integer 'remember the source node
Dim NN As Integer 'remember the being considered Next Neighbor's indexN
Dim NP(15) As Integer 'Number of paths. remember the number of paths respect to the same end node as the index.
Dim EndNode(15) As Integer 'Corresponding to NP(15), remember the 'end of the path, duplicate ends will only count once.

PathNum = 0 'set the number of path as 0
For i = 0 To 15 'set all elements of the current path as -1
    str(i) = -1
    NP(i) = 0
    EndNode(i) = -1
Next i

saveStartNode = startNode.IndexN 'remember the start node's indexN
lenP = 0 'set the current path's length as 0
str(lenP) = startNode.IndexN 'set the first node indexN as the start node
Node(startNode.IndexN).Direction = 1

Dict = Node(startNode.IndexN).Direction 'set the direction of the 'current node as the start node's direction

While Node(saveStartNode).Direction < 5 'after all of the start 'node's directions have been considered, the program end
    For i = Dict To 4 'there are at most 4 directions for each node
```

A. Source code: verify the base case of Theorem 3.3.3

NN = NextNeib(startNode.IndexN, i)

'computing the current node's ith neighbor
If Node(NN).Direction = 0 And lenP < PLen Then 'If it's

'neighbor is OK, then add the node to the path and go on.
lenP = lenP + 1 'remember the length of the path
str(lenP) = Node(NN).IndexN 'add the node to the path
Node(startNode.IndexN).Direction = i

'remember the direction for its father
startNode.IndexN = NN

'set it as the start node for the next step
startNode.Direction = 0

'set the current node's direction as 0
Dict = 1 'to be computing from its first direction
Exit For 'continue to consider its neighbor
Elself i = 4 Or lenP = PLen Then 'if all of the current node's

'directions has been considered or the length is right
If lenP = PLen Then

'the length is right, then print the path to the given array
For j = 0 To lenP 'record the current path.

Path(PathNum, j) = str(j)
Next j
PathNum = PathNum + 1 'number of paths increase 1
NP(str(lenP)) = NP(str(lenP)) + 1 'corresponding end node's path number increase 1
EndNode(str(lenP)) = startNode.IndexN

End If
If lenP > 0 Then 'if it is not the start node, then untreat

'again, otherwise, change the start node's direction we need
'to go back and try another direction, until we have tried

'all directions, so as to obtain every possible path.
Node(str(lenP)).Direction = 0
A. Source code: verify the base case of Theorem 3.3.3

\begin{verbatim}
'change its direction back to 0
str(lenP) = -1
lenP = lenP - 1 'no live neighbor, untreat one step
startNode.IndexN = str(lenP)
startNode.Direction = Node(str(lenP)).Direction
While startNode.Direction = 4 And lenP > 0 'If it untreat to the first node, then we should stop it.
    Node(str(lenP)).Direction = 0
    str(lenP) = -1
    lenP = lenP - 1
    startNode.IndexN = str(lenP)
    startNode.Direction = Node(str(lenP)).Direction
Wend
Dict = startNode.Direction + 1 'next time, we can't try the 'direction that has already tried. So, change the direction 'now.
End If
If lenP = 0 Then 'otherwise,change the start node's direction
    Node(saveStartNode).Direction = _
        Node(saveStartNode).Direction + 1
    Dict = Node(saveStartNode).Direction
End If
Exit For
End If
Next i
Wend

' Print #1, "End nodes are: "

totalEnds = 0
\end{verbatim}
While PathEnds(saveStartNode, totalEnds) <> -1 'to avoid overwrite
'the previous saved ends (different path length)
totalEnds = totalEnds + 1
Wend

For i = 0 To 15
    If NP(i) > 0 Then
        Print #1, i, NP(i)
        Print #1, i,
        PathEnds(saveStartNode, totalEnds) = i
        totalEnds = totalEnds + 1
    End If
Next i

PaNu = PathNum
'The following code writing each path to the file
    For i = 0 To PathNum - 1
        ss = ""
        For j = 0 To PLen
            ss = ss & Path(i, j) & " "
        Next j
        ss = i & " " & ss
        Print #1, ss
    Next i
End Sub
Private Sub FindPathEdgeFaulty(startNode As nodeDef, PLen As Integer)
Dim str(15) As Integer 'remember the index of the current path
Dim ss As String 'used to print information to file
Dim i As Integer, j As Integer
Dim lenP As Integer 'remember the current considering path's length
Dim Dict As Integer 'remember the direction of the current 'considering node
Dim PathNum As Integer 'remember the number of paths
Dim saveStartNode As Integer 'remember the source node
Dim NN As Integer
'remember the being considered Next Neighbor's indexN
Dim NP(15) As Integer
'Number of paths. Remember the number of paths respect to the same 'end node as the index.
Dim EndNode(15) As Integer 'Corresponding to NP(15), remember the 'end of the path, duplicate ends will only count once.

PathNum = 0 'set the number of path as 0
For i = 0 To 15 'set all elements of the current path as -1
str(i) = -1
NP(i) = 0
EndNode(i) = -1
Next i

saveStartNode = startNode.IndexN 'remember the start node's indexN
lenP = 0 'set the current path's length as 0
str(lenP) = startNode.IndexN 'set the first node indexN as the start node
Node(startNode.IndexN).Direction = 1

Dict = Node(startNode.IndexN).Direction 'set the direction of the
'current node as the start node's direction

While Node(saveStartNode).Direction < 5 'after all of the start
  'node's directions have been considered, the program end
  If Node(saveStartNode).BDirection(Node(saveStartNode).Direction)_
    Then 'the edge to it's neighbor should be available
      For i = Dict To 4 'there are at most 4 directions for each node
        NN = NextNeib(startNode.IndexN, i)
          'computing the current node's ith neighbor

        If Node(NN).BDirection(i) And Node(NN).Direction = 0 And_
          lenP < PLen Then 'If it's neighbor is OK, then add the
            'node to the path and go on.
          lenP = lenP + 1 'remember the length of the path has been add
          str(lenP) = Node(NN).IndexN 'add the node to the path
          Node(startNode.IndexN).Direction = i
            'remember the direction for its father
          startNode.IndexN = NN
            'set it as the start node for the next step
          startNode.Direction = 0 'set the current node's direction as 0
          Dict = 1 'to be computing from its first direction
        Exit For 'continue to consider its neighbor
      ElseIf i = 4 Or lenP = PLen Then 'if all of the current node's
        'directions has been considered or the length is right
        If lenP = PLen Then 'the length is right, then print the path
          'to the given array
          For j = 0 To lenP
            'record the current path.
            Path(PathNum, j) = str(j)
          Next j
          PathNum = PathNum + 1 'number of paths increase 1
          NP(str(lenP)) = NP(str(lenP)) + 1
'corresponding end node's path number increase 1

End If

If lenP > 0 Then ' if it is not the start node, then untreat again, otherwise, change the start node's direction we need to go back and try another direction, until we have tried all directions, so as to obtain every possible path.

Node(str(lenP)).Direction = 0

'change its direction back to 0

str(lenP) = -1
lenP = lenP - 1 'no live neighbor, untreat one step
startNode.IndexN = str(lenP)
startNode.Direction = Node(str(lenP)).Direction

While startNode.Direction = 4 And lenP > 0

'If it untreat to the first node, then we should stop it.

Node(str(lenP)).Direction = 0
str(lenP) = -1
lenP = lenP - 1

startNode.IndexN = str(lenP)
startNode.Direction = Node(str(lenP)).Direction

Wend

Dict = startNode.Direction + 1

'next time, we can't try the direction that has already tried. So, change the direction now.

End If

If lenP = 0 Then 'otherwise, change the start node's direction

Node(saveStartNode).Direction =

Node(saveStartNode).Direction + 1

Dict = Node(saveStartNode).Direction

End If

Exit For

End If
Next i
Else
    Node(saveStartNode).Direction = Node(saveStartNode).Direction + 1
End If
Wend

' Print #1, "End nodes are:"

totalEnds = 0

While PathEnds(saveStartNode, totalEnds) <> -1
    'to avoid overwrite the previous saved ends (different path length)
    totalEnds = totalEnds + 1
Wend

For i = 0 To 15
    If NP(i) > 0 Then
        ' Print #1, i, NP(i)
        ' Print #1, i,
        PathEnds(saveStartNode, totalEnds) = i
        totalEnds = totalEnds + 1
    End If
Next i

' Print #1,
' Print #1, "---------------------------------------------"
' Print #1, "Total: ", totalEnds
' Print #1,

PaNu = PathNum

' The following code writing each path to the file
' For i = 0 To PathNum - 1
A. Source code: verify the base case of Theorem 3.3.3

A. Source code: verify the base case of Theorem 3.3.3

' ss = ""
' For j = 0 To PLen
'    ss = ss & Path(i, j) & " "
' Next j
' ss = i & " " & ss
' Print #1, ss
' Next i

End Sub

Private Sub SortPathEndsO ' sort the ends in increasing order
    Dim i As Integer, j As Integer, k As Integer, h As Integer
    Dim remLast As Integer, remCur As Integer
    i = 0
    j = 0
    k = 0
    h = 0

    For i = 0 To 15
        If PathEnds(i, 0) <> -1 Then
            remCur = 0
            remLast = PathEnds(i, 0)
            For j = 0 To 15

                If remCur = -1 Or remLast = -1 Then
                    j = 16
                Else
                    remCur = PathEnds(i, j)
                    k = j - 1
                    While remCur < remLast And remCur <> -1
                        PathEnds(i, k + 1) = remLast
                    End While
                End If
            Next j
        End If
    Next i
A. Source code: verify the base case of Theorem 3.3.3

\[ \text{PathEnds}(i, k) = \text{remCur} \]
\[ k = k - 1 \]
\[ \text{If } k < 0 \text{ Then} \]
\[ \text{remLast} = -1 \]
\[ \text{Else} \]
\[ \text{remLast} = \text{PathEnds}(i, k) \]
\[ \text{End If} \]
\[ \text{Wend} \]
\[ \text{remLast} = \text{PathEnds}(i, j) \]
\[ \text{End If} \]
\[ \text{Next } j \]
\[ \text{End If} \]
\[ \text{Next } i \]
\[ \text{End Sub} \]

Private Sub Command1_Click()
    MsgBox "Please contact Mr Yonghong Xiang for more information: ", _
    & vbCrLf & "E-Mail: yh_xiang@hotmail.com" _
    & vbCrLf & "2008-11-18." 
End Sub

Private Sub Command24_Click()
    Unload Me
End Sub

Private Sub Form_Load()
    Addr = "C:\"
End Sub

Private Sub FVOneFZero_Click()
    Dim i As Integer, j As Integer, t As Integer
Dim startNode As nodeDef
Dim PathLengthTwelve As Integer, PathLengthThirteen As Integer

For i = 0 To 15 'initialize PathEnds array.
    For j = 0 To 15
        PathEnds(i, j) = -1
    Next j
Next i

totalEnds = 0 ' Initialize totalEnds

'we will only check two path length which is enough to prove our 'theorem.
PathLengthTwelve = 12
PathLengthThirteen = 13

Me.MousePointer = vbHourglass
Open Addr & "FVOneZero.txt" For Append As #1
'to write the result in a file.

For i = 0 To 15 'initialize PathEnds array.
    For j = 0 To 15
        PathEnds(i, j) = -1
    Next j
Next i

totalEnds = 0

expl = "Faulty nodes: 0. Path length: " & PathLengthTwelve & " and ".
       & PathLengthThirteen & "."
Print #1, expl
A. Source code: verify the base case of Theorem 3.3.3

InitNode
Node(0).Direction = 5

For i = 1 To 15
    startNode.IndexN = i
    startNode.Direction = 1
    If Node(i).Direction = 0 Then
        expl = "Start node: " & startNode.IndexN
        Print #1, expl
        FindPath startNode, PathLengthTwelve
        InitNode
        Node(0).Direction = 5
    End If
Next i

' expl = "Faulty nodes: 0 and 1. Path length: " & PathLengthElev & "."
' Print #1, expl

InitNode
Node(0).Direction = 5

For i = 1 To 15
    startNode.IndexN = i
    startNode.Direction = 1
    If Node(i).Direction = 0 Then
        expl = "Start node: " & startNode.IndexN
        Print #1, expl
        FindPath startNode, PathLengthThirteen
        InitNode
        Node(0).Direction = 5
End If

Next i

SortPathEnds 'sort all the end nodes increasingly

For i = 0 To 15
' to print out all ends that has reached from one node on length 10
'and 11.

expl = "Start node: " & i & " has end nodes: "

For j = 0 To 15
If PathEnds(i, j) = -1 Then
Exit For
Else
expl = expl & PathEnds(i, j) & "; "
End If
Next j

Print #1, expl & " Total: " & j
Next i

Print #1,
Close #1

MsgBox "Congratulation! Case: 1 faulty node, and path length ", 
& PathLengthTwelve & " and " & PathLengthThirteen
Me.MousePointer = 1
End Sub

Private Sub FVZero_Click()

Dim i As Integer, j As Integer, k As Integer, t As Integer
Dim NN As Integer
Dim startNode As nodeDef

Dim PathLengthFourteen As Integer, PathLengthFifteen As Integer
Dim FE1L(12) As Integer, FE1R(12) As Integer
dim FEOL As Integer, FEOR As Integer

PathLengthFourteen = 14
PathLengthFifteen = 15

Me.MousePointer = vbHourglass
Open Addr & "FVZero.txt" For Append As #1
' to write the result in a file.

FEOL = 0:FEOR = 1
FE1L(0) = 0:FE1R(0) = 3
FE1L(1) = 0:FE1R(1) = 4
FE1L(2) = 2:FE1R(2) = 3
FE1L(3) = 3:FE1R(3) = 7
FE1L(4) = 4:FE1R(4) = 5
FE1L(5) = 4:FE1R(5) = 8
FE1L(6) = 5:FE1R(6) = 6
FE1L(7) = 6:FE1R(7) = 7
FE1L(8) = 7:FE1R(8) = 11
FE1L(9) = 8:FE1R(9) = 9
FE1L(10) = 8:FE1R(10) = 11
FE1L(11) = 10:FE1R(11) = 11

For t = 0 To 11

InitNode ' set back all nodes to its original value.
A. Source code: verify the base case of Theorem 3.3.3

For i = 1 To 4
'set faulty edges, so that when we call the function FindPath,
'we can decide whether to go on some direction by its direction
'boolean value.
    NN = NextNeib(FE1L(t), i)
    If FE1R(t) = NN Then
        If i = 1 Or i = 3 Then ' horizontal direction
            Node(FE1L(t)).BDirection(i) = False
            Node(FE1R(t)).BDirection(4 - i) = False
        ElseIf i = 2 Or i = 4 Then ' vertical direction
            Node(FE1L(t)).BDirection(i) = False
            Node(FE1R(t)).BDirection(6 - i) = False
    End If
    End If
    Next i
Node(O).BDirection(1) = False
Node(1).BDirection(3) = False

For i = 0 To 15 ' initialize PathEnds array.
    For j = 0 To 15
        PathEnds(i, j) = -1
    Next j
    Next i
Next i

totalEnds = 0

expl = "Faulty edge: (0, 1) and (" & FE1L(t) & ", " & FE1R(t) & ")
' Path length: " & PathLengthFourteen & " and ":
& PathLengthFifteen & "."
Print #1, expl
A. Source code: verify the base case of Theorem 3.3.3

For i = 0 To 15
    startNode.IndexN = i
    startNode.Direction = 1
    If Node(i).Direction = 0 Then
        expl = "Start node: " & startNode.IndexN
        Print #1, expl
        FindPathEdgeFaulty startNode, PathLengthFourteen
    End If
Next i

InitNode ' set back all nodes to its original value.
For k = 1 To 4 ' set faulty edges, so that when we call the ' function FindPath, we can decide whether to go on some ' direction by its direction boolean value.
    NN = NextNeib(FE1L(t), k)
    If FE1R(t) = NN Then
        If k = 1 Or k = 3 Then ' horizontal direction
            Node(FE1L(t)).BDirection(k) = False
            Node(FE1R(t)).BDirection(4 - k) = False
        ElseIf k = 2 Or k = 4 Then ' vertical direction
            Node(FE1L(t)).BDirection(k) = False
            Node(FE1R(t)).BDirection(6 - k) = False
        End If
    End If
Next k
Node(O).BDirection(1) = False
Node(1).BDirection(3) = False

End If
Next i

InitNode ' set back all nodes to its original value.
For k = 1 To 4
'set faulty edges, so that when we call the function FindPath, 'we can decide whether to go on some direction by its direction 'boolean value.

NN = NextNeib(FEIL(t), k)
If FEIR(t) = NN Then
    If k = 1 Or k = 3 Then ' horizontal direction
        Node(FEIL(t)).BDirection(k) = False
        Node(FEIR(t)).BDirection(4 - k) = False
    ElseIf k = 2 Or k = 4 Then ' vertical direction
        Node(FEIL(t)).BDirection(k) = False
        Node(FEIR(t)).BDirection(6 - k) = False
End If
End If
Next k
Node(0).BDirection(1) = False
Node(1).BDirection(3) = False

For i = 0 To 15
    startNode.IndexN = i
    startNode.Direction = 1
    If Node(i).Direction = 0 Then
        expl = "Start node: " & startNode.IndexN
        Print #1, expl
        FindPathEdgeFaulty startNode, PathLengthFifteen
    End If
End For

InitNode ' set back all nodes to its original value.
For k = 1 To 4 ' set faulty edges, so that when we call the 'function FindPath, we can decide whether to go on some 'direction by its direction boolean value.
    NN = NextNeib(FEIL(t), k)
    If FEIR(t) = NN Then

If \( k = 1 \) Or \( k = 3 \) Then ' horizontal direction
\[
\text{Node(FElL(t)).BDirection(k) = False}
\]
\[
\text{Node(FElR(t)).BDirection(4 - k) = False}
\]
ElseIf \( k = 2 \) Or \( k = 4 \) Then ' vertical direction
\[
\text{Node(FElL(t)).BDirection(k) = False}
\]
\[
\text{Node(FElR(t)).BDirection(6 - k) = False}
\]
End If
End If
Next k
Node(0).BDirection(1) = False
Node(1).BDirection(3) = False
End If
Next i

SortPathEnds ' sort all the end nodes increasingly

For \( i = 0 \) To 15 'to print out all ends that has reached from one node 'on length 10 and 11.
exp1 = "Start node: " & i & " has end nodes: "
For \( j = 0 \) To 15
If PathEnds(i, j) = -1 Then
Exit For
Else
exp1 = exp1 & PathEnds(i, j) & "; "
End If
Next j
Print #1, exp1 & " Total: " & j
Next i

Print #1,
A. Source code: verify the base case of Theorem 3.3.3

Next t

Close #1

MsgBox "Congratulation! Case: two faulty edges, and path length ",
      & PathLengthFourteen & " and " & PathLengthFifteen
Me.MousePointer = 1
End Sub

Private Sub FVOne_Click()

    Dim i As Integer, j As Integer, k As Integer, t As Integer

    Dim startNode As nodeDef

    Dim PathLengthTwelve As Integer, PathLengthThirteen As Integer
    Dim FEL(5) As Integer, FER(5) As Integer

    PathLengthTwelve = 12
    PathLengthThirteen = 13

    Me.MousePointer = vbHourglass
    Open Addr & "FVOne.txt" For Append As #1
    'to write the result in a file.

    FEL(0) = 1:FER(0) = 2
    FEL(1) = 1:FER(1) = 5
    FEL(2) = 2:FER(2) = 6
    FEL(3) = 5:FER(3) = 6
FEL(4) = 6:FER(4) = 10

For t = 0 To 4 'consider the above five cases

InitNode ' set back all nodes to its original value.
For i = 1 To 4
'set faulty edges, so that when we call the function FindPath, we can
'decide whether to go on some direction by its direction boolean
'value.
    NN = NextNeib(FEL(t), i)
    If FER(t) = NN Then
        If i = 1 Or i = 3 Then ' horizontal direction
            Node(FEL(t)).BDirection(i) = False
            Node(FER(t)).BDirection(4 - i) = False
        ElseIf i = 2 Or i = 4 Then ' vertical direction
            Node(FEL(t)).BDirection(i) = False
            Node(FER(t)).BDirection(6 - i) = False
        End If
    End If
Next i
Node(O).Direction = 5

For i = 0 To 15 'initialize PathEnds array.
    For j = 0 To 15
        PathEnds(i, j) = -1
    Next j
Next i

totalEnds = 0

expl = "Faulty nodes: 0. Faulty edge: (" & FEL(t) & ", " & FER(t)_
A. Source code: verify the base case of Theorem 3.3.3

& "). Path length: " & PathLengthTwelve & " and ",
& PathLengthThirteen & "."
Print #1, expl

For i = 1 To 15
    startNode.IndexN = i
    startNode.Direction = 1
    If Node(i).Direction = 0 Then
        expl = "Start node: " & startNode.IndexN
        Print #1, expl
        FindPathEdgeFaulty startNode, PathLengthTwelve
    InitNode ' set back all nodes to its original value.
    For k = 1 To 4 'set faulty edges, so that when we call the
    'function FindPath, we can decide whether to go on some
    'direction by its direction boolean value.
        NN = NextNeib(FEL(t), k)
        If FER(t) = NN Then
            If k = 1 Or k = 3 Then ' horizontal direction
                Node(FEL(t)).BDirection(k) = False
                Node(FER(t)).BDirection(4 - k) = False
            ElseIf k = 2 Or k = 4 Then ' vertical direction
                Node(FEL(t)).BDirection(k) = False
                Node(FER(t)).BDirection(6 - k) = False
            End If
        End If
    Next k

    Node(0).Direction = 5
End If
Next i
A. Source code: verify the base case of Theorem 3.3.3

InitNode ' set back all nodes to its original value.
For k = 1 To 4 ' set faulty edges, so that when we call the function 'FindPath, we can decide whether to go on some direction by its 'direction boolean value.

NN = NextNeib(FEL(t), k)
If FER(t) = NN Then
    If k = 1 Or k = 3 Then ' horizontal direction
        Node(FEL(t)).BDirection(k) = False
        Node(FER(t)).BDirection(4 - k) = False
    ElseIf k = 2 Or k = 4 Then ' vertical direction
        Node(FEL(t)).BDirection(k) = False
        Node(FER(t)).BDirection(6 - k) = False
    End If
End If
Next k
Node(O).Direction = 5

For i = 1 To 15
    startNode.IndexN = i
    startNode.Direction = 1
    If Node(i).Direction = 0 Then
        ' expl = "Start node: " & startNode.IndexN
        ' Print #1, expl
        FindPathEdgeFaulty startNode, PathLengthThirteen
        InitNode ' set back all nodes to its original value.
        For k = 1 To 4 ' set faulty edges, so that when we call the ' function FindPath, we can decide whether to go on some ' direction by its direction boolean value.
            NN = NextNeib(FEL(t), k)
If FER(t) = NN Then
    If k = 1 Or k = 3 Then ' horizontal direction
        Node(FEL(t)).BDirection(k) = False
        Node(FER(t)).BDirection(4 - k) = False
    ElseIf k = 2 Or k = 4 Then ' vertical direction
        Node(FEL(t)).BDirection(k) = False
        Node(FER(t)).BDirection(6 - k) = False
    End If
    End If
    Next k
    Node(0).Direction = 5
End If
Next i

SortPathEnds ' sort all the end nodes increasingly

For i = 0 To 15 ' to print out all ends that has reached from one node
    exp1 = "Start node: " & i & " has end nodes: "
    For j = 0 To 15
        If PathEnds(i, j) = -1 Then
            Exit For
        Else
            exp1 = exp1 & PathEnds(i, j) & "; "
        End If
    Next j
    Print #1, exp1 & " Total: " & j
Next i

Print #1,
Next t
Close #1

MsgBox "Congratulation! Case: 1 faulty node and one faulty edge," & " and path length " & PathLengthTwelve & " and " & PathLengthThirteen
Me.MousePointer = 1

End Sub

Private Sub FVTwo_Click()
' there are only 4 cases to consider by the symmetric of Q_{2,k}
' they are: (0, 1), (0, 2), (0, 5), (0, 10)
' We suppose node 0 is faulty, if there is at least one faulty node.

Dim i As Integer, j As Integer, t As Integer
Dim StartNode As nodeDef
Dim F2(15) As Integer ' remember the second faulty node's index

Dim PathLengthElev As Integer, PathLengthTen As Integer

For i = 0 To 15 ' initialize PathEnds array.
    For j = 0 To 15
        PathEnds(i, j) = -1
    Next j
Next i

totalEnds = 0 ' Initialize totalEnds

' we will only check two path length which is enough to prove our theorem.
PathLengthElev = 11
PathLengthTen = 10

Me.MousePointer = vbHourglass
Open Addr & "FTwo.txt" For Append As #1
' to write the result in a file.

F2(0) = 1:F2(1) = 2
F2(2) = 5:F2(3) = 6
F2(4) = 10

For t = 0 To 4

For i = 0 To 15 ' initialize PathEnds array.
    For j = 0 To 15
        PathEnds(i, j) = -1
    Next j
Next i

totalEnds = 0

expl = "Faulty nodes: 0 and " & F2(t) & ". Path length: ",
    & PathLengthTen & " and " & PathLengthElev & "."
Print #1, expl

InitNode
Node(0).Direction = 5
Node(F2(t)).Direction = 5

For i = 1 To 15
    If i <> F2(t) Then
        startDate.IndexN = i
A. Source code: verify the base case of Theorem 3.3.3

startNode.Direction = 1
If Node(i).Direction = 0 Then
  expl = "Start node: " & startNode.IndexN
  Print #1, expl
FindPath startNode, PathLengthTen
InitNode
Node(0).Direction = 5
Node(F2(t)).Direction = 5
End If
End If
Next i

'expl = "Faulty nodes: 0 and 1. Path length: " & PathLengthElev & "."
'Print #1, expl

InitNode
Node(0).Direction = 5
Node(F2(t)).Direction = 5

For i = 1 To 15
  If i <> F2(t) Then
    startNode.IndexN = i
    startNode.Direction = 1
    If Node(i).Direction = 0 Then
      expl = "Start node: " & startNode.IndexN
      Print #1, expl
      FindPath startNode, PathLengthElev
      InitNode
      Node(0).Direction = 5
      Node(F2(t)).Direction = 5
    End If
  End If
Next i
A. Source code: verify the base case of Theorem 3.3.3

End If
Next i

SortPathEnds ' sort all the end nodes increasingly

For i = 0 To 15 'to print out all ends that has reached from one node
'on length 10 and 11.
    expl = "Start node: " & i & " has end nodes: "
    For j = 0 To 15
        If PathEnds(i, j) = -1 Then
            Exit For
        Else
            expl = expl & PathEnds(i, j) & "; "
        End If
    Next j
    Print #1, expl & " Total: " & j
Next i

Print #1,

Next t

Close #1

MsgBox "Congratulation! We have done for case: 2 faulty nodes,"_
& " and path length " & PathLengthTen & " and " & PathLengthElev
Me.MousePointer = 1

End Sub

Private Function NextNeib(NodeNum As Integer, Dirt As Integer)
A. Source code: verify the base case of Theorem 3.3.3

If Dirt = Rt Then
    If NodeNum Mod 4 = 3 Then
        NextNeib = NodeNum - 3
    Else
        NextNeib = NodeNum + 1
    End If
ElseIf Dirt = Lf Then
    If NodeNum Mod 4 = 0 Then
        NextNeib = NodeNum + 3
    Else
        NextNeib = NodeNum - 1
    End If
ElseIf Dirt = Dn Then
    NextNeib = (NodeNum + 4) Mod 16
Else
    NextNeib = (16 + NodeNum - 4) Mod 16
End If
End Function
Index

(n, k)-star graph, 17, 114
  1-edge, 17
  i-edge, 17
D-node, 142
P-node, 142
k-ary n-cube, 15, 26, 64, 85
k-connected, 10
k-container, 11
k-edge-fault-tolerant bipancyclic, 12
k-edge-fault-tolerant edge-bipancyclic, 12
k-edge-fault-tolerant hamiltonian laceable, 11
k-edge-fault-tolerant strongly hamiltonian laceable, 11
k-pairwise disjoint paths, 4
k-regular, 9
k-vertex-connected, 10
k*-connected, 11
k*-container, 11
m-panconnected, 12
n-dimensional augmented cube, 18
  complement edge, 19
  hypercube edge, 19
n-star graph, 16
k-edge-fault-tolerant hyper-hamiltonian laceable, 11
adjacent, 9
almost-hamiltonian, 11
almost-hamiltonian-connected, 11
almost-pancyclic, 12
augmented k-ary n-cube, 83–85
  (≤ i, −1)-edge, 85
  (≤ i, +1)-edge, 85
  (i, +1)-edge, 85
  (i, −1)-edge, 85
bipanconnected, 12
bipancyclic, 12
bipartite, 9
bipartition, 9
Cayley graph, 88
column, 28
conditional fault assumption (CFA), 21
connectivity, 10
containers, 23
cycle, 10
degree, 9
diameter, 10
dimension, 15
distance, 10
edge, 9
edge-bipancyclic, 12
edge-pancyclic, 12
fault, 27
graph, 9
grid, 13
hamiltonian, 11
hamiltonian cycle, 10
hamiltonian laceable, 11
hamiltonian path, 10
hamiltonian-connected, 11
Hamming distance, 12
healthy, 28
hyper hamiltonian laceable, 11
hypercube, 3, 14, 15
join, 28
Lee distance, 13
Lee weight, 13
length, 10
link, 9
many-to-many disjoint paths, 5, 25
many-to-one disjoint paths, 4
mesh, 13
neighborhood, 9
node, 9
node-symmetric, 9
node-to-node disjoint paths, 4
node-to-set disjoint paths, 4
one-to-many disjoint paths, 4, 24
one-to-one disjoint paths, 4, 23
panconnected, 12
parity, 27
even, 27
odd, 27
partition, 15
path, 10
progressively shortened, 65
residual node, 11
row, 28
set-to-set disjoint paths, 4
spanning connectivity, 12
strongly hamiltonian laceable, 11
subgraph, 9
super spanning connected, 12
torus, 14
row-torus, 14
vertex, 9
vertex cut, 10
vertex-bipancyclic, 12
vertex-connectivity, 10
vertex-pancyclic, 12
vertex-symmetric, 9, 65