Properties and Algorithms of the KCube Graphs

Li Zhao

Department of Computer Science

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science

Faculty of Mathematics and Science, Brock University

St. Catharines, Ontario

Acknowledgements

I would like to express my heartfelt gratitude to my supervisor, Dr. Ke Qiu , whose patience and suggestions on an interesting thesis topic, as well as his academic experience, have been invaluable to me. I genuinely thank him for his general and financial support over the past two years. He has not only guided me in the correct direction for my research but has also provided some brilliant ideas in the development of my thesis.

I would also like to thank my committee members, including Dr. M. Winter, Dr. S. Houghten and Dr. B. Farzad, for their thoughtful and detailed comments, feedback, and suggestions.

As a graduate student at Brock University, I am deeply grateful to the Faculty of Mathematics and Science for offering a rich and fertile environment to study and explore new ideas. Brock University is also gratefully acknowledged.

Finally, I would like to thank my parents, Guohua Zhao and Fenglian Cui for being a constant source of support. It is thanks to my father that I first became interested in the subject of science when I was a child.

Thank you very much.

Li Zhao

Abstract

The KCube interconnection topology was first introduced in 2010. The KCube graph is a compound graph of a Kautz digraph and hypercubes. Compared with the attractive Kautz digraph and well known hypercube graph, the KCube graph could accommodate as many nodes as possible for a given indegree (and outdegree) and the diameter of interconnection networks. However, there are few algorithms designed for the KCube graph. In this thesis, we will concentrate on finding graph theoretical properties of the KCube graph and designing parallel algorithms that run on this network.

We will explore several topological properties, such as bipartiteness, Hamiltonianicity, and symmetry property. These properties for the KCube graph are very useful to develop efficient algorithms on this network. We will then study the KCube network from the algorithmic point of view, and will give an improved routing algorithm. In addition, we will present two optimal broadcasting algorithms. They are fundamental algorithms to many applications. A literature review of the state of the art network designs in relation to the KCube network as well as some open problems in this field will also be given.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The motivation to investigate parallel computation is that it is an evolution of serial computation. Traditionally, serial computation is a single computer having a single central processing unit (CPU). A problem can be broken into a discrete series of instructions. Instructions can be executed by a CPU sequentially. In computing, there are many large and complex problems that are impossible or impractical to be solved by a single computer, especially given that even the most complex computer has an upper limit of processors and computer memory. Such problems include galaxy information, weather forecasting, biomedical analyses, speech recognition, the management of huge knowledge bases and so on. One way out of this impasse is to provide more computer resources in parallel computation. More processors and computer memory per problem will shorten the completion time. In a parallel computer, multiple processors cooperate to solve a computational problem in a portion of the time required by one processor. In general, there are two important aspects of parallel computation, namely, parallel computational models and parallel algorithms. The

relation between parallel computational models and parallel algorithms is a parallel algorithm solves a particular problem on a parallel computational model for which the parallel algorithms are designed. We have two major computational models, namely, shared memory parallel machines and interconnection networks, depending on how the processors communicate with each other.

This chapter will introduce parallel computer architectures and the reasons for our interest in the KCube. We will present illustrations of shared memory parallel machines and interconnection networks. Important aspects of good interconnection networks and some popular topologies will then be laid out. We will then explain a number of parameters that researchers use to analyze and measure parallel algorithms. Finally, we will give an overview and the organization of this thesis.

1.1 Parallel Computer Architectures

There are diverse methods to characterize parallel computer architectures. We can categorize computers into following four classifications according to the interaction between instruction streams and data streams [17].

- **Single Instruction, Single Data Stream (SISD)**

  In a SISD computer, there is only one instruction stream being executed by the CPU and only one data stream being used as an input. Typical SISD type computers are older generation mainframes and minicomputers.

- **Multiple Instruction, Single Data Stream (MISD)**

  In a MISD computer, there are multiple processors operating on a single data

stream independently through separate instruction streams.

- **Single Instruction, Multiple Data Stream (SIMD)**

  A SIMD computer has multiple processors which execute the same instructions. Each processor can operate on a different data element. The key characteristic of a SIMD computer on data streams is that the operations can be performed in parallel on each element of a large regular data structure. All processors are controlled by a central control unit. All the major parallel models in this thesis are SIMD computers.

- **Multiple Instruction, Multiple Data Stream (MIMD)**

  In an MIMD computer, all processors execute different instruction streams and work with different data streams. Execution can be synchronous or asynchronous. MIMD machines are the most powerful type of the parallel computers.

1.2 Shared Memory Parallel Machines

One of the most important categories of parallel machines is shared memory parallel machines. A shared memory parallel machine consists of identical processors and a common memory which they access in parallel. The communication of a shared memory machine occurs implicitly in the same way that a conventional computer accesses instructions (i.e., loads and stores). All processors can run independently and share the shared memory variables. One processor can write a variable in a shared memory location that will be visible to all other processors. A shared memory parallel

machine is also called as a Parallel Random Access Machine (PRAM). Examples of PRAM are commonly represented today by Symmetric Multiprocessor Machines (SMP) and CC-UMA-Cache Coherent (UMA). Cache coherent means if one processor updates a location in the shared memory, all the other processors know about the update. Cache coherency is accomplished at the hardware level.

Cooperation and coordination among all the processors in PRAM are accomplished simultaneously by reading and writing the shared variables in the shared memory through a memory access unit (MAU).

1.3 Interconnection Networks

The previous section introduced communication among processors via a shared memory. In this section, we will: give an explanation of what an interconnection network is; give a brief overview of why new interconnection networks continue to be proposed by listing the most important aspects of interconnection networks; explain why de Bruin and Kautz networks are excellent candidates for interconnection network designs; give formal definitions of interconnection networks; list several popular interconnection networks; and finally; list important criteria for evaluating network topologies. The difference between a shared memory parallel machine and an interconnection network is that there is no a common memory for processors to share in an interconnection network. Instead, an equal portion of the common memory is distributed to all the processors. The communication among all the processors in an interconnection network is via topological networks. Two processors which are

directly connected by a link are said to be neighbours. In a complete network with N processors (N means the number of processors in a network in this thesis), each processor has $N-1$ neighbours and can send a datum to any of its neighbours and receive a datum from any of its neighbours through a two-way link [1]. It is not only too expensive to build so many links for all but the smallest values of N but also infeasible to lay out the number of links connecting all the processors without too many crossings. As a consequence, there is a huge volume of research papers focusing on a number of aspects of topological network design, taking into consideration that; (1) interconnection networks in parallel computation usually possess a regular pattern; (2) most topological networks attempt to minimize the diameter; (3) the topological structure of interconnection networks has superior mathematical properties which have close fundamental relationships with the communication patterns of important parallel algorithms [14]; (4) vertex symmetry is also a desirable attribute of an efficient interconnection network design [29]. This property makes any vertex look same in the network. Moreover, symmetric networks allow for identical processors at every vertex with identical routing algorithms. It's very useful to design efficient algorithms that exploit the symmetric structure of the network. Many well-known interconnection networks, such as complete networks, rings, tori, hypercubes, cube-connected cycles, star graphs, and pancake graphs are examples of such vertex-symmetric networks. Most of them belong to the class of Caley graphs which are connected graphs constructed from a group and a set of generators [2, 7, 26, 27, 33, 39, 40]; (5) another valuable physical topology for interconnection networks is to have the largest number of vertices for a fixed degree $\triangle$ and diameter D [29]. As the demand for data

processing expands year by year, it is desirable to design interconnection networks that will allow for the maximum number of processors with the most efficient and the lowest-cost possible combination of degree and diameter while maintaining simple routings, such as those in de Bruijn and Kautz networks. A de Bruijn network, which can provide the shortest distance between clusters running different parts of an application [10], was chosen for JPLs 8096-node multiprocessor.

When attempting to create a very large multiprocessor system [29], one of the important aforementioned aspects is the fifth aspect: a network which has the largest number of vertices for a fixed degree and diameter. This is what led us to focus on the KCube: the KCube is a very large multiprocessor system. We looked deeply into de Bruijn and Kautz networks because our research was to explore the topological properties of, and to design parallel algorithms on, the KCube network, and the KCube is a compound graph of a Kautz digraph and hypercubes, with the Kautz digraph being a subdigraph of the de Bruijn digraph. Therefore, before we explain our research into the KCube, we need to examine the de Bruijn and Kautz networks upon which the KCube is based.

In fact, de Bruijn and Kautz networks are not vertex symmetric [9, 29]. However, they have very elegant properties, such as having an optimal number of nodes (for a fixed value of D or $\triangle$ bounded by Moore Bound [28]), easy routings, and an optimal fault-tolerance [12]. General upper bounds called Moore bounds for the order of such graphs and digraphs are attainable only for certain special graphs and digraphs. Those graphs focus on finding better (tighter) upper bounds for the maximum possible number of vertices by giving the degree/diameter.

Formally, we can use an undirected graph to describe an interconnection network. Given an undirected graph $G = (V, E)$, where each processor P_i is located at the vertex v_i and there exists a direct communication link between two processors P_i and P_j if and only if $(v_i, v_j) \in E$. V is a set of vertices and E is a set of edges. Interconnection networks, in general, can be classified into direct or indirect schemes [3, 14, 32]. In this thesis, we will use the terms "processor"and "node", "interconnection network"and "graph"interchangeably. Next, we will give a brief description of some popular interconnection networks. These networks have been proposed, built, and used as commercial computers. From now on, we will use N to denote the number of processors in the following interconnection networks.

Complete Graph: The complete graph is the most powerful network. In a complete graph K_N, each of the processors is adjacent to the remaining $N-1$ processors. A complete graph is also called a **Clique**.

Linear Array: The linear array is the simplest way to connect N processors, $P_0, P_1, ..., P_{N-1}$. In this network, all N processors form a one-dimensional array. Each processor P_i $(0 < i < N-1)$ is adjacent to its two neighbours P_{i-1} and P_{i+1}. The first node P_0 is adjacent to P_1 and the last node P_{N-1} to P_{N-2}. Both of them have only one neighbour. If we connect P_0 and P_{N-1}, we get a network called **Ring**. In this case, every node has two neighbours.

Two-Dimensional Array: A network is obtained by arranging the N processors into an $r \times s$ two dimensional array. The processor in row i and column j is denoted by P_{ij}, where $0 \leq i \leq r-1$ and $0 \leq j \leq s-1$. Each processor P_{ij} has two-way communication links to its four neighbours $P_{(i+1)j}$, $P_{(i-1)j}$, $P_{i(j+1)}$, and $P_{i(j-1)}$

if they exist. Processors on the boundary rows and columns have fewer than four neighbours. This network is also known as **Mesh**. A multi-dimensional mesh can be defined similarly. Such a network is called a d-dimensional mesh, where $d \geq 2$. Each processor in a d-dimensional mesh is adjacent to its $2 \times d$ neighbours, except the processors on the boundary.

Tree: In a tree network, the processors form a complete binary tree. Such a tree has d levels, numbered 0 to $d-1$, and $N = 2^d - 1$ nodes, each of which is a processor. Each processor at level i is connected by a two-way communication line to its parent at level $i+1$ and to its two children at level $i-1$. The root processor (at level $d-1$) has no parent, and the leaves (all of which are at level 0) have no children.

Pyramid: A one-dimensional pyramid parallel computer is obtained by adding two-way links connecting processors at the same level in a binary tree, thus forming a linear array at each level. This concept can be extended to higher dimensions. For example, a two-dimensional pyramid consists of $(4^{d+1} - 1)/3$ processors distributed among $d+1$ levels. All processors at the same level are connected to form a mesh. There are 4^d processors at level 0 (also called the base), arranged in a $2^d \times 2^d$ mesh. There is only one processor at level d (also called the apex). In general, a processor at level i, in addition to being connected to its four neighbours at the same level, has connections to four children at level $i-1$ (provided that $i \geq 1$) and to one parent at level $i+1$ (provided that $i \leq d-1$).

Shuffle Exchange: Let N processors $P_0, P_1, ..., P_{N-1}$ be available, where N is a power of 2. In the perfect shuffle interconnection, a one-way communication line links P_i to P_j, where

$$j = \begin{cases} 2i, & \text{for } 0 \leq i \leq N/2 - 1 \\ 2i + 1 - N, & \text{for } N/2 \leq i \leq N - 1 \end{cases}$$

.

Equivalently, the binary representation of j is obtained by cyclically shifting that of i one position to the left. An alternative representation of the perfect shuffle interconnection is given as a mapping from the set of processors to itself. This representation explains the origin of the network's name: When a deck of playing cards is split into two piles of equal size, a perfect shuffle is obtained by interleaving the cards in the two piles. If the directions on the one-way links are reversed, we obtain the perfect unshuffle connection. The links are undirected (i.e., are two-way communication lines). It is known as the shuffle-unshuffle network. Finally, two-way lines connecting every even-numbered processor to its successor are added to the network. These connections are called exchange links. A network with the shuffle, unshuffle, and exchange links is called a shuffle exchange network.

A number of criteria are used to evaluate network topologies. We will now introduce some of them and then use them to analyze some of the networks described above.

Definition 1. *The **degree** of a processor is the number of neighbours of this processor.*

Definition 2. *The **distance** between two processors P_i and P_j is the number of links on the shortest path from P_i to P_j; the **diameter** of the network is the maximum distance among any two arbitrary processors.*

Table 1.1 shows the interconnection networks already defined and the ones to be

Table 1.1: Interconnection Networks and their Degrees and Diameters

Interconnection network	Degree	Diameter	Precise Diameter
Linear Array	2	$O(N)$	$N-1$
$r \times s$ Mesh	4	$O(max(r,s))$	$(r-1)+(s-1)$
Tree	3	$O(\log N)$	$2\lfloor \log N \rfloor$
Pyramid	9	$O(\log N)$	$O(\log N)$
Shuffle-Exchange	3	$O(\log N)$	$2\log N$-1
Hypercube(n-cube)	$\log N$	$O(\log N)$	$\log N$
$B(d,k)$	d	k	k
$K(d,k)$	d	k	k
$KC(m,k)$	$m+1$	$O(mk)$	$m(k+1)+1$

defined in Chapter 2.

Definition 3. *The **connectivity** of a graph G with N nodes is $N-1$ if G is the complete graph and otherwise is the minimum number of nodes of G whose deletion results in a disconnected graph.*

The larger connectivity is, the better.

Definition 4. *The **Bisection width** is the minimum number of arcs that must be removed to partition the network into two equal halves.*

The larger the bisection width the better.

Definition 5. *A graph G is **f-fault tolerant** if, whenever f or less than f nodes are deleted from G, the remaining graph is still connected. The **fault tolerance** of the graph G is the maximum f for which it is f-fault tolerant.*

The difference between connectivity and fault tolerance is 1.

Definition 6. *A graph is **regular** if and only if all nodes in this graph have the same degree.*

We can divide interconnection networks into two models, based on how many neighbours one can communicate with in one time unit (similar to PRAM) [1] :

- **single-port** (weak) model, in each unit of time a processor is only allowed to send data to or receive data from one of its neighbours.
- **all-port** (strong) model, the processor can communicate with one or more of its neighbours simultaneously.

Unless specified otherwise, all interconnection networks in this thesis are considered to be the single-port model.

1.4 Analyzing Parallel Algorithms

There are different ways to evaluate a parallel algorithm. In a sequential algorithm, running time is the major measurement. In a parallel algorithm, evaluation consists of the running time, the number of the processors, the speedup, the slowdown, and the cost. The following measurement is from [1].

The **running time** of a parallel algorithm is defined as the time required by the algorithm to solve a computational problem on a parallel computer. The running time is measured by counting the number of consecutive elementary steps performed by the algorithm in a worst-case scenario. There are two different types of elementary steps in parallel algorithms:

- **A computational step** is a basic arithmetic or logical operation performed on one or two data within a processor. A parallel computer performs multiple op-

erations in a single step in order to solve a problem efficiently. These operations are comprised of adding, comparing, swapping, etc.

- **A routing step** is used by an algorithm to route a constant size datum between the source node and the destination node across the shared memory or interconnection network.

We make an assumption that each computational step or each routing step takes a constant number of time units. The total number of the steps is the running time of a parallel algorithm. We use a function $t(N)$ to denote the running time of a parallel algorithm of the input size N.

Another aspect to measure the performance of a parallel algorithm is the **number of processors** which is a function of the size of the input. To compare different parallel algorithms for a given problem with the same running time, fewer processors are preferred. We use $p(N)$ to denote the number of processors used by a parallel algorithm to solve a problem of size N.

The **cost** $c(N)$ of a parallel algorithm is defined as $c(N) = p(N) \times t(N)$. If the lower bound for solving the problem is $\Omega(f(N))$ in a sequential computation, and the cost of a parallel algorithm is $O(f(N))$ in a parallel computation, we say that this parallel algorithm is cost optimal.

It is significant to balance cost and performance, since a parallel computer with more processors will be more expensive to build.

Speedup: The primary reason for using parallel algorithms is to speed up sequential computation. This is measured by a ratio known as the speedup, defined

as follows: let t_s denote the worst case running time of the fastest known sequential algorithm for a problem, and let t_p denote the worst case running time of the parallel algorithm using p processors. Then, the speedup provided by the parallel algorithm is

$$Speedup = \frac{t_s}{t_p}.$$

A good parallel algorithm is one for which this ratio is large.

Speedup Folklore Theorem: For a given computational problem, the speedup provided by a parallel algorithm using p processors, over the fastest possible sequential algorithm for the problem, is at most equal to p.

Slowdown is the effect on the running time of reducing the number of the processors on a parallel computer.

Slowdown Folklore Theorem: If a certain computation can be performed with p processors in time t_p and with q processors in time t_q, where $q \leq p$, then $t_p \leq t_q \leq t_p + pt_p/q$.

1.5 Organization of the Thesis

Three of the commonly studied interconnection networks in the past are de Bruijn, Kautz, and hypercubes. The advantages of de Bruijn networks (such as Koorde [25]), and Kautz networks (such as lightwave networks [31, 34]) are that they have an optimal number of nodes (for small value of $\triangle$ and diameter D), easy routings and an optimal fault-tolerance [29]. The advantages of hypercube networks (such as the

Cosmic Cube, the Intel iPSC, the NCUBE) are that they possess strong connectivity, which means that several node-disjoint paths exist between any two nodes; regularity, which means that every node has the same degree; and symmetry, which means that there exists an automorphism for any pair of nodes to map one node to another [10].

The main disadvantages of de Bruijn and Kautz diagraphs are that they are not vertex symmetric, and therefore cannot employ identical routing algorithms and other algorithms, such as sorting algorithms and so on, at every vertex [29]. The main disadvantage of hypercubes is that the degree of each node in a hypercube increases as $\log N$, where N is the total number of nodes. This property could make its use prohibitive for large N [10].

All of these networks attempt to reduce diameter as far as possible and employ simple routing algorithms. Most importantly, to design an efficient physical topology, to reduce the amount of signal loss, network topology should attempt to design a digraph with as many nodes as possible for a given indegree (and outdegree) and diameter [31, 34, 38]. For example, a de Bruijn topology can handle more nodes than shufflenet with similar performance measures, and a Kautz topology can have more nodes than a de Bruijn topology with the same degree and diameter. In addition, a KCube topology can accommodate more nodes than a Kautz topology with the same degree and diameter, which makes it worthy of investigating a KCube topology.

Among state of the art schemes, a dBCube network is closest to the KCube network. The dBCube network was proposed in 1993 as a compound graph of a de Bruijn graph and hypercubes. The dBCube possesses the advantages of maintaining equal connectivity for all nodes in the network, but, it suffers from large diameter

when the size of the network increases [10]. Because of the limitations of the above mentioned networks, the KCube graph was proposed in 2010 as a novel architecture for interconnection networks. The KCube is a compound graph of a Kautz digraph and hypercubes which replaces each node on a Kautz digraph with a hypercube as a cluster.

The KCube network employs hypercube topology as a basic cluster, connects many such clusters using a Kautz digraph, and maintains node connectivity to be the same for all nodes. The size of the KCube network as a regular network can be easily extended by increments of a cluster size. Each cluster utilizing hypercube topology allows easy embedding of existing parallel algorithms, while the Kautz digraph, which possesses easy routings and an optimal fault-tolerance, provides the shortest distance between clusters running different parts of an application. On the other hand, the KCube is not a vertex-symmetric network which makes it difficult for us to design identical routing algorithms at every node. Another disadvantage of the KCube is that the degree of the Kautz digraph must be equal to the number of nodes in a hypercube which is constrained against an arbitrary hypercube size. As a proposed network, some topological properties and fundamental algorithms for the KCube have not been exploited yet. They include bipartiteness, Hamiltonianicity, symmetry property, improved routing algorithms and broadcasting algorithms. We will explore the KCube network from both the graph theoretical and the algorithmic points of views in this thesis. We will present the following:

1. a literature review of the KCube;
2. related interconnection networks and their properties for de Bruijn graph, Kautz digraph, and Hypercube;
3. new topological properties for the KCube and an improved routing algorithm that provides a smaller upper bound on the diameter of the KCube;
4. two broadcasting algorithms for the KCube;
5. summary;

This thesis is organized as follows. Chapter 2 will present a literature review of the KCube and related interconnection networks. We will also define various problems to be studied in this thesis. In Chapter 3, some graph theoretical properties will be discussed and an improved routing algorithm will be presented. We will design two broadcasting algorithms for the KCube in Chapter 4. In Chapter 5, we will summarize our work and offer suggestions for future work.

Chapter 2

Literature Review of the KCube Graph

2.1 Introduction

In this chapter, we will provide a formal definition of the KCube and will review the primary research of Guo et al. [20] into the KCube. Before we do this, however, we will give formal definitions of graph properties and a brief background of the predecessors of the KCube in the development of some interconnection networks - the de Bruijn digraph, Kautz digraph, and hypercube. We will provide formal definitions, topological properties, routings and uses of these networks. We will then look at the dBCube, the compound graph which most closely resembles the KCube, before proceeding to the KCube itself.

The de Bruijn, Kautz, and hypercube networks have been much studied because of the properties they have, which have made them good candidates for interconnection

networks of parallel computers (see [5, 36]), such as Koorde [25], D2B [18], Tapestry, Pastry, Lightwave [34] and FissioneE [30]. Koorde and D2B are based on de Bruijn network, Tapestry and Pastry are based on hypercube topology and Lighwave and FissioneE networks are based on Kautz network.

2.2 Properties

Definition 7. *A graph $G = (V, E)$ is a **bipartite graph** if (1) $V = V_1 \bigcup V_2$ and $V_1 \bigcap V_2 = \emptyset$ and (2) $\forall (u, v) \in E$, $u \in V_1$ and $v \in V_2$ or $u \in V_2$ and $v \in V_1$. In other words, no two nodes in V_1 are adjacent and no two nodes in V_2 are adjacent. The complete bipartite graph with partitions of equal size $\mid V_1 \mid = n$ and $\mid V_2 \mid = n$, is denoted $K_{n,n}$ [42].*

Definition 8. *An **Eulerian Cycle** in an undirected graph is a cycle that uses each edge exactly once. A connected graph has an Eulerian cycle if and only if all vertices have even degree [42].*

Definition 9. *A **Hamiltonian Cycle** is a spanning cycle in a graph (a cycle through every vertex). The circumference of a graph is the length of its longest cycle [42].*

Definition 10. *A **Hamiltonian path** is a spanning path [42].*

Definition 11. *An **Isomorphism** from G to H (another graph) is a bijection $f : V(G) \to V(H)$ such that $(v, w) \in E(G)$ if and only if $f(v, w) \in E(H)$. We say G is isomorphic to H, written $G \cong H$, if there is an isomorphism from G to H [42].*

Definition 12. *An **Automorphism** of G is a permutation of $V(G)$ that is an isomorphism from G to G [42].*

Definition 13. *A graph G is **vertex symmetric or (vertex-transitive)** if for every pair $v, w \in V(G)$, there is an automorphism that maps v to w [42].*

Definition 14. *The **line graph** of G, written $L(G)$, is the graph whose vertices are the edges of G, with $(e, f) \in E(L(G))$ when $e = (u, v)$ and $f = (v, w)$ in G (i.e., when e and f share a vertex) [42].*

2.3 de Bruijn Digraph

A **de Bruijn digraph** $B(d, k)$ of outdegree d and diameter k has as vertices the words of length k on an alphabet of d letters. The number of vertices in $B(d, k)$ is d^k. Vertex $x_1...x_k$ is joined by an arc to the vertices $x_2...x_k\alpha$ where α is any letter from the alphabet, $\alpha = 0, 1, ..., d-1$ [4]. For a given digraph G, we denote as UG the underlying graph associated to G (obtained by removing the orientation). The underlying de Bruijn graph will therefore be denoted as $UB(d, k)$.

Topological Properties

- Each k-dimensional de Bruijn digraph is the line digraph of the $(k-1)$-dimensional de Bruijn digraph with the same set of vertices [23].

- Each de Bruijn digraph is Eulerian and Hamiltonian. The Euler cycles and Hamiltonian cycles of these digraphs (equivalent to each other via the line digraph construction) are de Bruijn sequences. A binary de Bruijn sequence

of order k is a string of bits $d_i \in \{0,1\}$, $d = \{d_1, ..., d_2^n\}$ such that ever string of length k, $\{a_1, ..., a_n\} \in \{0,1\}^n$, occurs exactly once consecutively in d. Each $B(d,k)$ sequence has length d^k. There are $\frac{(d!)^{d^{k-1}}}{d^k}$ distinct de Bruijn sequences in $B(d,k)$. There are two distinct de Bruijn sequences which are 00010111 and 11101000. Each sequence can generate all the vertices [16]. For example: 00010111 = (000)10111 = 0(001)0111 = 00(010)111 = 000(101)11 = 0001(011)1 = 00010(111) = 00101(110) = 001011(100). The de Bruijn sequences $B(2,3)$ can be constructed by taking a Hamiltonian path of a 3-dimensional de Bruijn digraph over 2 symbols (or equivalently, an Eulerian cycle of a 2-dimensional de Bruijn digraph over 2 symbols) [16].

- A simple routing method in a de Bruijn network is as follows: Let $x = x_1...x_k$ be a source node and $y = y_1...y_k$ be a destination node. The routing path is $x_1...x_k \longrightarrow x_2...x_k y_1 \longrightarrow x_3...y_1 y_2 \longrightarrow \cdots \longrightarrow x_k y_1...x_{k-1} \longrightarrow y_1...y_k$ [4].

Uses

- Some grid network topologies are de Bruijn graphs [11]. A grid network is a computer network that has a number of (computer) systems connected in a grid topology. In a grid topology, each node is connected with two neighbors along one or more dimensions.

- The distributed hash table protocol Koorde [25] uses a de Bruijn graph.

- In bioinformatics, de Bruijn graphs are used for de novo assembly of (short) read sequences into a genome [13].

2.4 Kautz Digraph

A directed **Kautz digraph** [31], $K(d,k)$, has node indegree = outdegree = d and network diameter = k, where $d \geq 1$ and $k \geq 1$. The number of vertices in $K(d,k)$ is $d^k + d^{k-1}$. The vertex set is $\{x_k...x_i...x_1 \mid x_i \in \{0,1,...d\}$ and $x_i \neq x_{i+1}$ for all $1 \leq i \leq k\}$, where $x_k...x_i...x_1$ denotes a sequence. There is an arc from vertex $x_k x_{k-1}...x_1$ to vertex $x_{k-1}...x_1\alpha$ for each $\alpha \in \{0,1,...,d\} - x_1$. The underlying Kautz graph will be denoted as $UK(d,k)$. Figure 2.1 gives an example of $K(2,2)$.

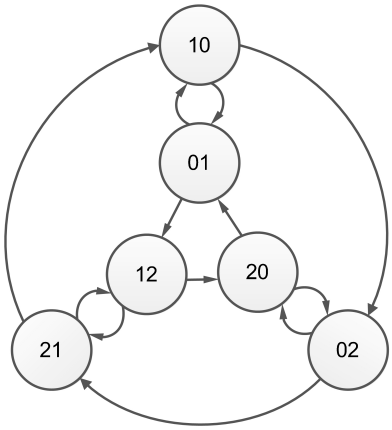


Figure 2.1: K(2,2)

Topological Properties

- For a fixed degree d and number of vertices $N = d^k + d^{k-1}$, the Kautz digraph has the smallest diameter of any possible directed graph with N vertices and degree d. A graph for a fixed degree d and diameter k, the maximum number of nodes N in the graph is the Moore bound [8] $1+d+d^2+...+d^k$. The Moore bound is not achievable for any non-trivial graph. The number of nodes in the Kautz digraph $K(d,k)$, $d^k + d^{k-1}$, is very close to the Moore bound. In fact, a Kautz digraph is the densest graph when the diameter is two. From the Moore bound,

it is easy to see the lower bound of the diameter of a graph with N nodes is $\lceil \log_d(N(d-1)+1) \rceil - 1$ and the diameter k of Kautz digraph $K(d,k)$ reaches the lower bound as $\lceil log_d(N(d-1)+1) \rceil - 1 = \lceil log_d((d^k + d^{k-1})(d-1)+1) \rceil - 1 = \lceil log_d(d^{k+1} - d^{k-1} + 1) \rceil - 1 = k$. Thus, the Kautz digraph has an optimal diameter. As far as the Moore bound is concerned, the best of the general known classes of networks are de Bruijn or Kautz networks [29, 30].

- All Kautz digraphs have Eulerian cycles (An Eulerian cycle is one which visits each edge exactly once. This result follows because Kautz digraphs have indegree equal to outdegree for each node) [41].

- All Kautz digraphs have Hamiltonian cycles [21].

- The Kautz digraph also has optimal fault tolerance [12, 30]. That is, Kautz digraph of degree d is d - connected (i.e., there are d node disjoint paths between any two nodes).

- For any two nodes $x = x_k...x_1$ and $y = y_k...y_1$, there is a shortest routing path of length k in $K(d,k)$ from x to y, given by $x = x_k...x_1 \longrightarrow x_{k-1}x_{k-2}...x_1y_k \longrightarrow x_{k-2}x_{k-3}...y_ky_{k-1} \longrightarrow \cdots \longrightarrow x_1y_k...y_2 \longrightarrow y = y_k...y_1 = y$ when $x_1 \neq y_k$ or a routing path of length $k-1$ given by $x = x_k...x_1 \longrightarrow x_{k-1}x_{k-2}...x_1y_{k-1} \longrightarrow x_{k-2}x_{k-3}...y_{k-1}y_{k-2} \longrightarrow \cdots \longrightarrow x_1y_{k-1}...y_1 = y = y_k...y_1 = y$ when $x_1 = y_k$ [31].

Uses

The Kautz digraph has been used as a network topology for connecting processors in high-performance computing [3], fault-tolerant computing applications [30], and as

attractive logical topologies in multihop lightwave networks [31, 34].

2.5 Hypercube

An m-dimensional hypercube is also known as an m-cube [1]. Let N be 2^m for some $m \geq 0$ and label all processors as $P_0, P_1, ..., P_{N-1}$. In an m-cube, for P_i, let $y_{m-1}y_{m-2}...y_1y_0$ be the binary representation of i, where $0 \leq i < N$. The processors P_i and P_j are adjacent if and only if the binary representations of the indices i and j differ in exactly one bit. Figure 2.2 shows $H(m)$, for $m = 0, 1, 2, 3, 4$.

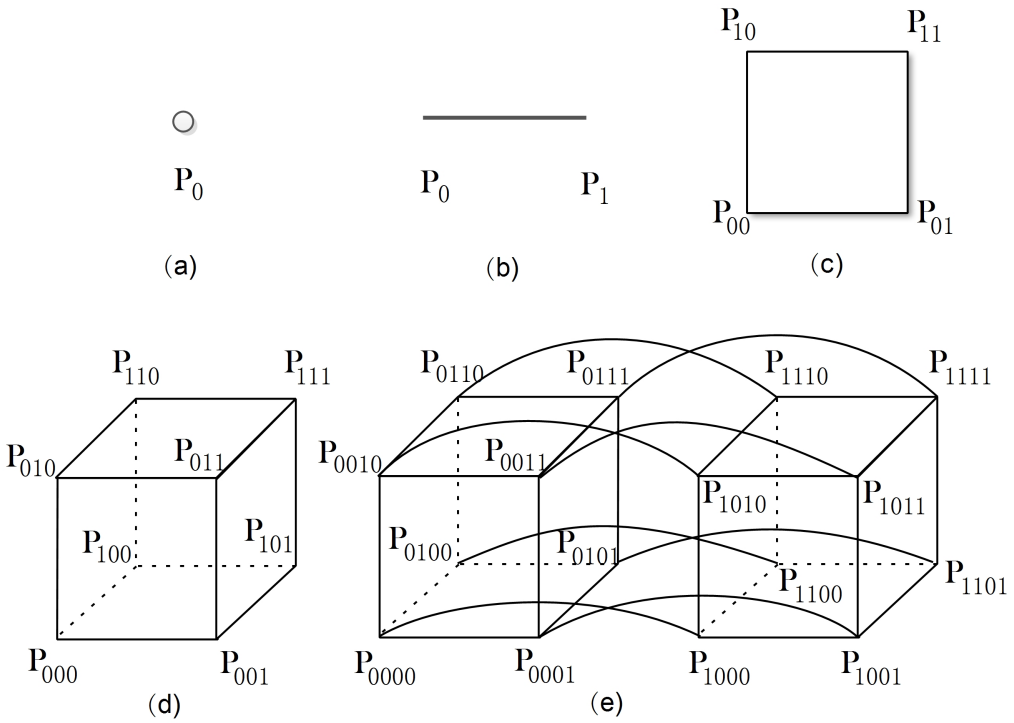


Figure 2.2: A Hypercube Interconnection Network: (a) $m = 0$; (b) $m = 1$; (c) $m = 2$; (d) $m = 3$; (e) $m = 4$

Topological Properties

The following hypercube properties are from [35].

- The m-cube can be constructed recursively from two lower dimensional $(m-1)$-cubes whose vertices are numbered likewise from 0 to $2^{m-1}-1$.

- There are m different ways of tearing an m-cube, i.e., of splitting it into two $(m-1)$-subcubes so that their respective vertices are connected in one-to-one fashion. Each different tearing corresponds to splitting the m-cube graph into two subgraphs: one whose node labels have an one in position i and one whose node labels have a zero in position i, $0 \leq i \leq m-1$.

- There are $m!2^m$ different ways in which the 2^m nodes of an m-cube can be labeled.

- Any two adjacent nodes A and B of an m-cube are such that the nodes adjacent to A and those adjacent to B are connected in an one-to-one fashion.

- There are no cycles of odd length in an m-cube.

- The minimum distance between the nodes A and B is equal to the number of bits that differ between A and B, i.e., the Hamming distance $h(A,B)$. The routing path between two nodes in an m-cube is to correct the positions in which A and B differ different bits.

  For example, let $A = 01011$ and $B = 10111$, one routing path is $01011 \longrightarrow 11011 \longrightarrow 10011 \longrightarrow 10111$.

- Let A, B be any two nodes and assume that $H(A,B) < m$. Then there are $H(A,B)$ parallel paths of length $H(A,B)$ between the nodes A and B.

- Let A, B be any two nodes of an m-cube and assume that $H(A, B) < m$. Then there are m parallel paths between A and B. Moreover, the length of each path is at most $H(A, B) + 2$.

2.6 dBCube

Definition 15. *Given two regular graphs G_1 and G_2, the compound graph $G_2(G_1)$ is obtained by replacing each node of G_2 by a copy of G_1 and replacing each link of G_2 by links which connects corresponding two copies of G_1 [20].*

Now, we will introduce the compound graph that is dBCube. A dBCube is a compound graph of a de Bruijn graph (G_2) and hypercubes (G_1). A dBCube is denoted as $dBC(c, d)$, where c is the number of hypercube clusters and d is the dimension of the hypercube. There are $c \times 2^d$ number of nodes in a $dBC(c, d)$ [10].

2.7 The KCube Network

The KCube combines a Kautz digraph G_2 and a hypercube G_1. It utilizes the hypercube topology as a unit cluster and connects all the hypercube clusters in the Kautz digraph. A constraint must be satisfied for the two graphs to comprise the KCube graph. The constraint is that the degree of the Kautz digraph G_2 must be the same as the number of nodes in a hypercube G_1. The outdegree and indegree of each node in $K(d, k)$ are d. The total degree of each node in $K(d, k)$ is $2d$. Each hypercube cluster has 2^m number of nodes. So, it is $d = 2^{m-1}$ in the KCube. The KCube is

denoted as $KC(m, d, k)$ where m is the hypercube dimension, d is the node outdegree of the Kautz digraph, and k is the diameter of the Kautz digraph [20]. For simplicity, we will use $KC(m, k)$ in this paper. Figure 2.3 shows the $K(2, 2)$ and $KC(2, 2)$. Any node in $KC(m, k)$ is labelled $\langle x = x_k...x_2x_1, y = y_m...y_2y_1\rangle$, where $x_k...x_2x_1$ is called the Kautz-part-label and $y_m...y_2y_1$ is called the hypercube-part-label. All nodes in $H(m)$ can be separated into two equal parts. The output nodes of $H(m)$ are the nodes $y_m...y_i...y_2y_1$, where

- $y_2y_1 = x_2x_1$ or $y_2y_1 = \bar{x_2}\bar{x_1}$.
- $y_i = 0$ or 1 for all $3 \leq i \leq m$.

The input nodes of $H(m)$ are the nodes $y_m...y_2y_1$, where

- $y_2y_1 = \bar{x_2}x_1$ or $y_2y_1 = x_2\bar{x_1}$.
- $y_i = 0$ or 1 for all $3 \leq i \leq m$.

The nodes whose last two bits are 00 or 11 are the output nodes of $H(m)$, and the nodes whose last two bits are 01 or 10 are the input nodes of $H(m)$. All nodes in $H(3)$ as shown in Figure 2.4 (a) are partitioned into two equal sets. The black nodes represent the output nodes, while the white nodes represent the input nodes, as shown in Figure 2.4 (b). The second precondition is that we further sort all the output nodes of $H(m)$ in the ascending order of the node labels and all the input nodes of $H(m)$ in the same way. For example, the output nodes of $H(m)$ are sorted in the order of 000, 011, 100, 111, and sort all the input nodes of $H(m)$ are sorted in the order of 001, 010, 101, 110, as shown in Figure 2.4 (b) [20].

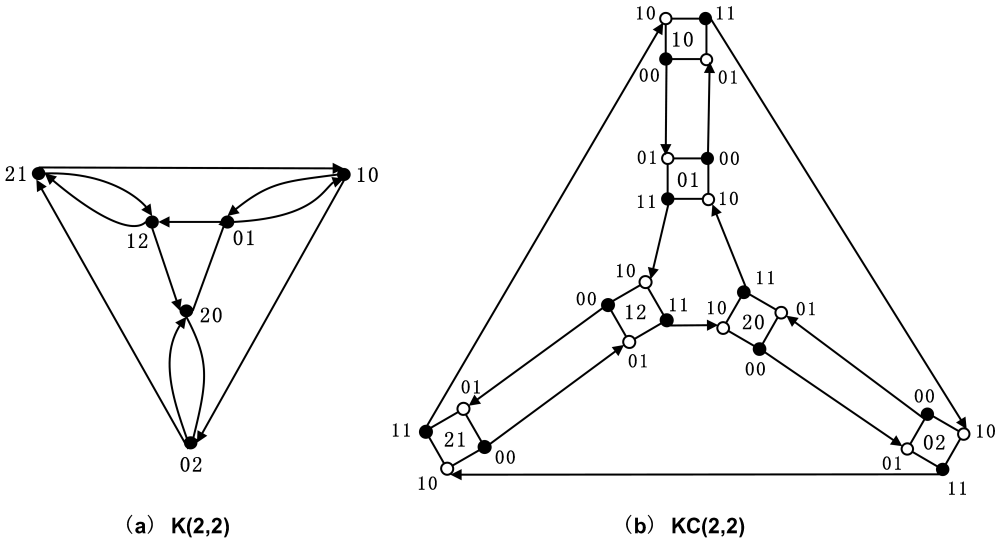


Figure 2.3: K(2,2) and KC(2,2)

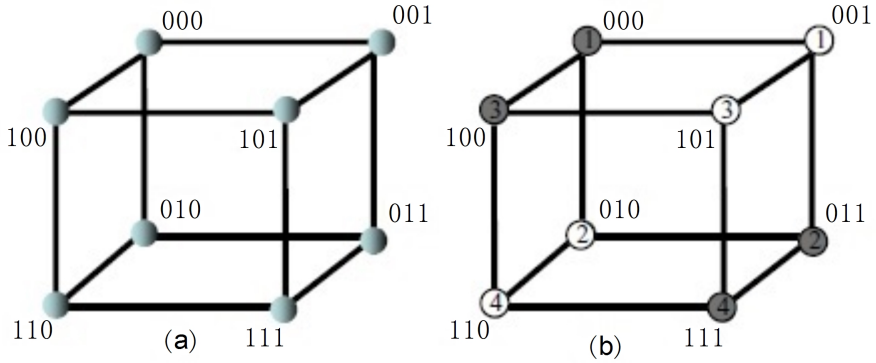


Figure 2.4: Partition and Sort of all Nodes in $H(3)$ [20]

Topological Properties

In a $KC(m,k)$, there are $2^{k(m-1)}+2^{(k-1)(m-1)}$ hypercube clusters and $2^{k(m-1)+m}+2^{k(m-1)+1}$ vertices [20].

Lemma 1. *The largest length of the shortest path between an output node and an input node in the same hypercube $H(m)$ is $m-1$ [20].*

Proof. In $H(m)$, the largest number of nodes which must be traversed in order to travel from any node $y_m...y_2y_1$ to the node$\bar{y_m}...\bar{y_2}\bar{y_1}$ is m. The length of the shortest

path between the node $y_m...y_2y_1$ and every node except the node $\bar{y_m}...\bar{y_2}\bar{y_1}$ is less than m. According to the splitting approaches, we can infer that the node $y_m...y_2y_1$ is included in the set of output nodes or the set of input nodes, together with the node $\bar{y_m}...\bar{y_2}\bar{y_1}$. Thus, the largest length of the shortest path between an output node and an input node in the same hypercube $H(m)$ is $m-1$. □

Theorem 1. *An upper bound on the diameter of* $KC(m,k)$ *is* $2m+(k-1)(m-1)+k = m(k+1)+1$ *[20].*

Proof. In $KC(m,k)$, the shortest path from an arbitrary node $\langle x,y\rangle$ to any node $\langle x',y'\rangle$ traverses at most $k+1$ hypercube clusters, including the source hypercube x, destination hypercube x', and other $k-1$ intermediate hypercubes. This is guaranteed by the diameter of $K(2^{m-1},k)$. In the source hypercube x, the largest length of a shortest path from the node $\langle x,y\rangle$ to any other node is less than or equal to m, and is only equal to m when the node $\langle x,y\rangle$ is an output node and the other node is $\langle x,\bar{y}\rangle$. For any intermediate hypercube along the shortest path from $\langle x,y\rangle$ to $\langle x',y'\rangle$, it receives a message from one of its input nodes and forwards the message to one of its output nodes within $m-1$ hops, as proved in Lemma 1. For the destination hypercube x', it receives a message from one of its input nodes and forwards this to the node $\langle x',y'\rangle$. The largest length of a shortest path from that input node to the node $\langle x',y'\rangle$ is less than or equal to m, and is only equal to m when that input node is $\langle x',\bar{y'}\rangle$ and the node $\langle x',y'\rangle$ is also an input node. In addition, the shortest path also traverses k remote links where each link connects a pair of hypercube clusters. Thus, a upper bound on the diameter of $KC(m,k)$ is $2m+(k-1)(m-1)+k = m(k+1)+1$,

hence Theorem 1 holds. □

Chapter 3

Topological Properties of the KCube Network

3.1 Introduction

In this chapter, we will give proofs that the KCube possesses three topological properties including bipartiteness, Hamiltonianicity for certain hypercube dimensions, and symmetry property. These properties are very useful in developing efficient parallel algorithms on the KCube network. Then, we will provide an improved routing algorithm that gives a smaller upper bound on the diameter of the KCube.

3.2 Topological Properties of the KCube

These properties are important because they affect how KCube algorithms are designed. We are interested in bipartiteness because it is a graph property that allows

us to partition all of the nodes on the KCube into two equal sets. We are interested in Hamiltonianicity because Hamiltonian cycles give us one way to sort all vertices on the KCube like those on a linear array. We are interested in the symmetry property because in any vertex symmetric network, all the vertices are identical. This makes it easier to design an identical routing algorithm on each vertex. We are interested in shortest path routing algorithms because the cost for the routing/communication between two nodes is proportional to the length of the routing path.

Theorem 2. *The KCube, as an undirected graph, is a bipartite graph.*

Proof. For each $H(m)$ used as a cluster, its bipartition is as follows:

$V_1 = \{$All nodes with even Hamming Weights$\}$,

$V_2 = \{$All nodes with odd Hamming Weights$\}$,

i.e.,

$V_1 = \{u_i \mid H(u_i)$ is even, $0 \leq i \leq 2^{m-1} - 1\}$,

$V_2 = \{v_j \mid H(v_j)$ is odd, $0 \leq j \leq 2^{m-1} - 1\}$.

With the above partition for each cluster, the nodes of the KCube are partitioned as follows:

$V_1' = \{\langle a, b\rangle | b \in V_1$, a is a Kautz label$\}$

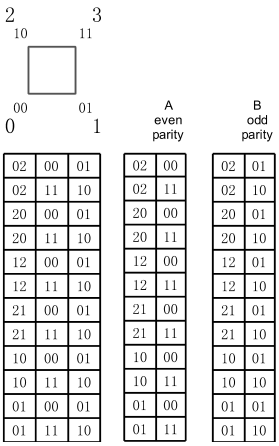
$V_2' = \{\langle c, d\rangle | d \in V_2$, c is a Kautz label$\}$

By the definition of the KCube, all output nodes are of the form:

$y_{m-2}...y_i...00$ or $y_{m-2}...y_i...11$, where, $y_i \in \{0, 1\}$, $i = 1, 2, ..., m-2$. All input nodes are of the form: $y_{m-2}...y_i...01$ or $y_{m-2}...y_i...10$, where, $y_i \in \{0, 1\}$, $i = 1, 2, ..., m-2$. For example, for $m = 3$, the sorted order of the output nodes are

000, 011, 100, 111, and the sorted order of the input nodes are 001, 010, 101, 110. Similarly, for $m = 4$, the sorted order of the output nodes are 0000, 0011, 0100, 0111, 1000, 1011, 1100, 1111, while the sorted order of the input nodes are 0001, 0010, 0101, 0110, 1001, 1010, 1101, 1110. It is easy to prove by induction that the output/input nodes in sorted order can be obtained by the sorted order of output/input nodes from $H(m)$ in $H(m+1)$. The sorted order of the output nodes in $H(m)$ are $u_1, u_2, ..., u_{2^{m-1}}$ and the sorted order of the input nodes in $H(m)$ are $v_1, v_2, ..., v_{2^{m-1}}$. Then, the sorted output nodes in $H(m+1)$ are $0u_1, 0u_2, ..., 0u_{2^{m-1}}$, $1u_1, 1u_2, ..., 1u_{2^{m-1}}$. The sorted input nodes in $H(m+1)$ are $0v_1, 0v_2, ..., 0v_{2^{m-1}}$, $1v_1, 1v_2, ..., 1v_{2^{m-1}}$. The initial condition: for $m = 2$, $u_1 = 00$, $u_2 = 11$, $v_1 = 01$, $v_2 = 10$. Therefore, we can see: $y_{m-2}...y_i...00$ is the i^{th} output node in $H(m)$. $y_{m-2}...y_i...01$ is the i^{th} input node in $H(m)$. $y_{m-2}...y_i...11$ is the j^{th} output node in $H(m)$. $y_{m-2}...y_i...10$ is the j^{th} input node in $H(m)$. It is equivalent to say that the i^{th} output node and the i^{th} input node have different parities. We use G that stands for Hamming distance in $H(m)$. For example, if $(\langle a, b\rangle, \langle c, d\rangle)$ is an edge in the $KC(m, k)$, where a, c are Kautz labels and b, d are hypercube labels, then $|G(b) - G(d)| = 1$. It is clear that it is impossible for any two nodes in $V_1^{'}$ $(V_2^{'})$ to be adjacent to each other because all of the hypercube labels in $V_1^{'}$ $(V_2^{'})$ have the same even/odd parity. Therefore, $KC(m, k)$ is a bipartite graph. □

Figure 3.1 shows $KC(2, 2)$ partitioned into two sets (A and B). The first column represents the nodes' Kautz-part-labels in the $KC(2, 2)$. The second and the third columns represent the nodes' hypercube-part-labels (even and odd parities) in the



02	00	01
02	11	10
20	00	01
20	11	10
12	00	01
12	11	10
21	00	01
21	11	10
10	00	01
10	11	10
01	00	01
01	11	10

02	00
02	11
20	00
20	11
12	00
12	11
21	00
21	11
10	00
10	11
01	00
01	11

02	01
02	10
20	01
20	10
12	01
12	10
21	01
21	10
10	01
10	10
01	01
01	10

Figure 3.1: Partition of KC(2,2)

$KC(2,2)$. The 2-cube is labelled as 00, 01, 10, 11. The 0, 1, 2, 3 are corresponding decimal numbers in the 2-cube. Each hypercube cluster in the $KC(2,2)$ has four vertices which are partitioned into two sets A and B. In set A, all nodes have even parities in the $KC(2,2)$. And, in the set B, all nodes have odd parities in the $KC(2,2)$.

Theorem 3. *$KC(1,k)$, for any k, is Hamiltonian.*

Proof. A Hamiltonian cycle of $KC(1,k)$ can be generalized as follows: A Hamiltonian cycle of $KC(1,k)$, for any k, on the Kautz-part-label can be constructed by Eulerian cycles of $K(1,k)$ digraph. An Eulerian cycle of $K(1,k)$ is guaranteed because the indegree and outdegree of each node is the same, namely, $d = 1$. The total degree of each node (both indegree and outdegree) will be an even number, namely, $2d = 2$. These two conditions guarantee Eulerian cycles of $K(1,k)$ for any k. If we replace each node in $K(1,k)$ with $H(1)$, we can have Hamiltonian cycles in $KC(1,k)$. This is because each node in $K(1,k)$ is replaced by an edge $H(1)$ with one input node and one output node where these two nodes follow the edge order of an Eulerian cycle

of $K(1,k)$ digraph. The edge order means a walk in a graph that is a sequence of vertices and edges, V_1, E_1, V_2, E_2,..., V_k, E_k, V_{k+1} such that the endpoints of edge E_i are V_i and V_{i+1}. If $V_1 = V_{k+1}$, the walk is a closed walk. A closed walk is to start and end at the same place. A successful walk in Knigsberg corresponds to a closed walk in the graph in which every edge is used exactly once. Such a closed walk in any graph that uses every edge exactly once is called an Eulerian cycle. In conclusion, we can use Eulerian cycles of $K(1,k)$ to get Hamiltonian cycles of $KC(1,k)$, for any k. □

Theorem 4. *$KC(2,k)$, for any k, is Hamiltonian.*

Proof. A Hamiltonian cycle of $KC(m,k)$, $m = 2$, can be generalized as follows: A Hamiltonian cycle of the $KC(2,k)$, for any k, on the Kautz-part-label can be constructed by an Eulerian cycle of $K(2,k)$ digraph. A connected digraph has Eulerian cycles if and only if every node's degree is even. In general, an Eulerian cycle is a digraph cycle which uses each digraph edge only once and a Kautz digraph has Eulerian cycles because the indegree and outdegree of each node is the same and the total degree of each node is even. $K(2,k)$ is guaranteed to have Eulerian cycles because the indegree and outdegree of each node is the same, namely, $d = 2$ and the total degree of each node (both indegree and outdegree) will be an even number, namely, $2d = 4$. These two conditions guarantee Eulerian cycles of $K(2,k)$ for any k. If we replace each node in $K(2,k)$ with a hypercube $H(2)$, we can have Hamiltonian cycles of $KC(2,k)$ for any k. This is because in the Eulerian cycle of $K(2,k)$, each node is replaced by two edges of $H(2)$, with each edge having one input node connected

to two output nodes. It is Hamiltonian as long as the input nodes and output nodes of any hypercube in $KC(2,k)$ follow the edge order of an Eulerian cycle of $K(2,k)$ digraph. In conclusion, we can use Eulerian cycles of $K(2,k)$ to get Hamiltonian cycles of $KC(2,k)$, for any k. □

For example, $KC(2,2)$ is a compound graph of $K(2,2)$ and $H(2)$. A Hamiltonian cycle of $KC(2,2)$ on the Kautz-part-label can be constructed by an Eulerian cycle of $K(2,2)$. The total degree of each node (both indegree and outdegree) in $K(2,2)$ is $2d = 4$. An Eulerian cycle of $K(2,2)$ traverses each edge in $K(2,2)$ only once. These edges are represented in Figure 3.2 (a) as $10 \to 01 \to 12 \to 21 \to 12 \to 20 \to 02 \to 20 \to 01 \to 10 \to 02 \to 21 \to 10$. If we replace each node in $K(2,2)$ with $H(2)$, we have Hamiltonian cycles of $KC(2,2)$. This is because in the Eulerian cycle of $K(2,2)$, each node is replaced by two edges of $H(2)$, with each edge having one input node connected to two output nodes (see Figure 3.3). Inside the $KC(2,2)$, all input nodes and output nodes of the six hypercube clusters follow the edge order of an Eulerian cycle of $K(2,2)$ digraph. The Hamiltonian cycle of $KC(2,2)$ is $\langle 10,10\rangle \to \langle 10,00\rangle \to \langle 01,01\rangle \to \langle 01,11\rangle \to \langle 12,10\rangle \to \langle 12,00\rangle \to \langle 21,01\rangle \to \langle 21,00\rangle \to \langle 12,01\rangle \to \langle 12,11\rangle \to \langle 20,10\rangle \to \langle 20,00\rangle \to \langle 02,01\rangle \to \langle 02,00\rangle \to \langle 20,01\rangle \to \langle 20,11\rangle \to \langle 01,10\rangle \to \langle 01,00\rangle \to \langle 10,01\rangle \to \langle 10,11\rangle \to \langle 02,10\rangle \to \langle 02,11\rangle \to \langle 21,10\rangle \to \langle 21,11\rangle$. The red lines in Figure 3.2 (b) represent the Hamiltonian cycle of $KC(2,2)$.

It is worth investigating in the future whether our proof for $KC(1,k)$ and $KC(2,k)$ also apply to $KC(m,k)$ for $m \geq 3$ or whether $KC(m,k)$ is Hamiltonian at all.

Proposition 1. *$KC(m,k)$ is regular but not vertex symmetric.*

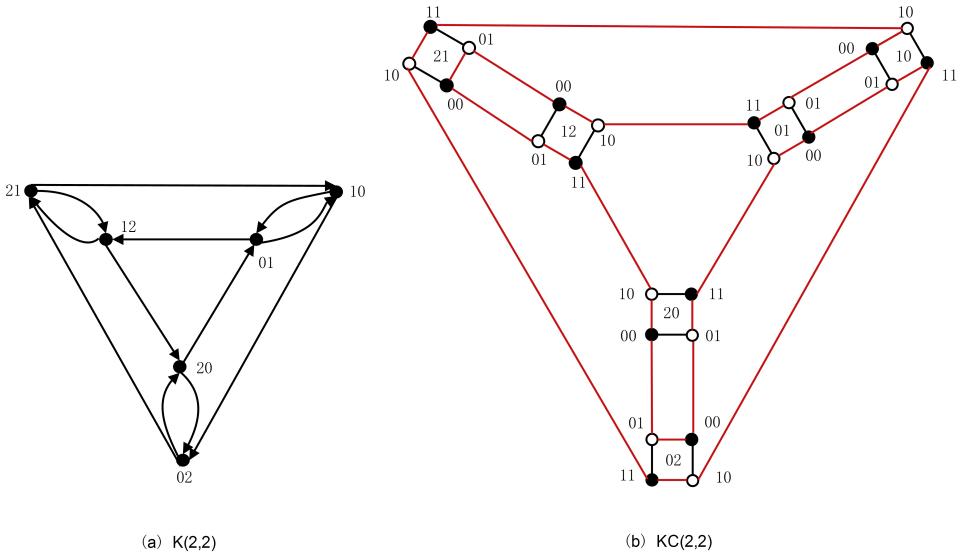


Figure 3.2: A Hamiltonian Cycle of KC(2,2)

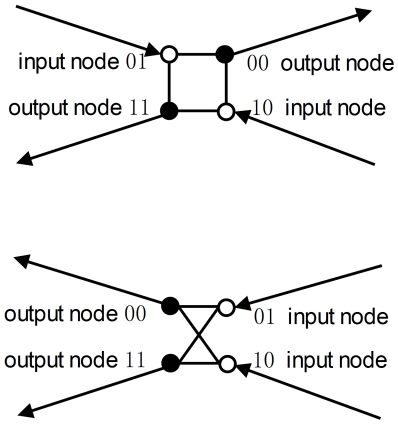


Figure 3.3: $H(2)$

Proof. The KCube is a regular graph because it is a compound graph of a Kautz digraph and hypercubes which are all regular. The KCube is not vertex symmetric because a graph is vertex symmetric if for any two vertices that there is an automorphism that maps u to v. In the $KC(2,2)$, certain nodes are on a cycle of length 6, while other nodes are on cycles of length 10. In fact, de Bruijn and Kautz graphs are not vertex symmetric [9, 29]. As a result, the KCube is not vertex symmetric.

□

3.3 Routing Problems

After proving the useful properties of the KCube, we want to investigate routing algorithms for the KCube, because an efficient routing algorithm leads to simpler and faster communication between any pair of nodes on a network. Guo et al.'s routing algorithm simply combined the Kautz routing algorithm and the hypercube routing algorithm, which allowed them to establish an upper bound on the diameter of the KCube which is $m(k+1)+1$ [20]. In the following, we develop an improved routing algorithm which provides a smaller upper bound on the diameter of the KCube.

Guo et al.'s routing algorithm is to combine the Kautz routing algorithm with the hypercube routing algorithm to the corresponding out-arc of the next Kautz node (which is represented on the KCube as a Kautz-part-label) and continues to a destination node in a hypercube cluster. However, their routing algorithm doesn't always give the shortest paths for all pairs of nodes. In their routing algorithm, there are $k+1$ hypercube clusters involved because the diameter of $K(d,k)$ is k. In our routing algorithm, we examine their routing algorithm as a possibility, but our work also explores other possible routing paths by adding an additional hypercube cluster along the routing paths. This enables us to determine other shorter routing paths and to establish a smaller upper bound on the diameter of the KCube which is $km+2$. This will involve at most $k+2$ hypercube clusters because the diameter of $K(d,k+1)$ is $k+1$. In the source hypercube cluster, there are 2^{m-1} output nodes besides the one

output node used by Guo et al. There are $(2^{m-1}-1)$ output nodes left, which means there are $(2^{m-1}-1)$ legal/possible routing paths if we add an additional hypercube cluster. We don't need to check all routing paths. We can do a little bit of analysis to choose the right output node which will result in shorter paths for some pairs of nodes in some cases. When each node in a Kautz digraph is replaced with a hypercube, the out-arcs (and the corresponding output nodes) and in-arcs (and the corresponding input nodes) are sorted on the hypercube clusters in a KCube in ascending order [20]. The out-arcs and in-arcs of any node in $K(2^{m-1},k)$ can be sorted using the following approach. One can infer from the definition of Kautz digraph that for a node $x_k...x_2x_1$, its out-arc to node $x_{k-1}...x_1\alpha$ for $\alpha \in \{0,1,...,d\} - \{x_1\}$ is denoted as the i^{th} out-arc. Here, i indicates the clockwise distance from x_k to α (if $x_1 \neq x_k$) or from x_k+1 to α (if $x_1 = x_k$) in a ring consisting of the values 0, 1,..., d in ascending order. It is worth noticing that the i^{th} out-arc of any node $x_k...x_2x_1$ is also the i^{th} in-arc of a corresponding node $x_{k-1}...x_1\alpha$. Thus, all 2^{m-1} out-arcs of each node can be sorted in ascending order, and all 2^{m-1} in-arcs of each node can be sorted in the same way. Moreover, first any given node x in $K(2^{m-1},k)$ is replaced by a hypercube $H(m)$. Second, the i^{th} out-arc of the node x is replaced by a remote arc between the i^{th} output node of a hypercube and the i^{th} input node of another hypercube. Here, the two hypercube clusters correspond to the head and tail of the i^{th} out-arc of the node x in $K(2^{m-1},k)$ [20]. When we analyze any pair of nodes, we use a graphical analogy of a ring that lists the values: 0, 1,..., d in ascending order in a clockwise direction which allows us to determine the three major routing paths among all possible routing paths. We map the second leftmost digit of the Kautz-

part-label in a source hypercube cluster to the leftmost digit on the Kautz-part-label in a destination hypercube cluster in a clockwise direction on the ring.

Algorithm 1. *An improved Routing Algorithm for $KC(m, k)$.*

We now present an improved routing algorithm. Guo et al. determine a routing algorithm in $KC(m, k)$. However, as they state in [20] their routing algorithm does not always derive the shortest paths for all pairs of nodes. For example, in $KC(2, 2)$, consider the routing from node $\langle 21, 11\rangle$ to node $\langle 20, 01\rangle$. According to their routing algorithm, the path is $\langle 21, 11\rangle \rightarrow \langle 21, 10\rangle \rightarrow \langle 21, 00\rangle \rightarrow \langle 12, 01\rangle \rightarrow \langle 12, 11\rangle \rightarrow \langle 20, 10\rangle \rightarrow \langle 20, 00\rangle \rightarrow \langle 20, 01\rangle$. This path has seven hops. There is another path which takes 5 hops: $\langle 21, 11\rangle \rightarrow \langle 10, 10\rangle \rightarrow \langle 10, 11\rangle \rightarrow \langle 02, 10\rangle \rightarrow \langle 02, 00\rangle \rightarrow \langle 20, 01\rangle$.

For this reason, we have proposed an improved routing algorithm which identifies the shorter paths for some pairs of nodes in some cases. we label two hypercube clusters x and x'. Their routing path from x to x' traverses at most $k + 1$ hypercube clusters, including the source hypercube cluster x and the destination hypercube cluster x', as well as other $k - 1$ intermediate hypercube clusters.

However, there are shorter paths between the source node $\langle x, y\rangle$ and the destination node $\langle x', y'\rangle$ if we can select an additional hypercube cluster v along the routing path. This can involve a maximum of $k + 2$ hypercube clusters, including the source hypercube cluster x and the destination hypercube cluster x', as well as other k intermediate hypercube clusters in certain cases. Now, we will present an approach for how the source node x selects the next hypercube cluster v which is either on the k remote-arc path or the $k + 1$ remote-arc paths. A link (arc) connecting two nodes in

the same cluster is called a local link, while a link connecting two nodes in different clusters is called a remote link [20]. The connection topology inside each cluster is a hypercube, while the interconnection topology at the level of clusters is a Kautz digraph. In this paper, we use the terms "link"and "arc"interchangeably.

The k remote-arc path used by Guo et al. travels from the source hypercube cluster x to the destination hypercube cluster x'. It will go through $k+1$ hypercube clusters at most because the diameter of $K(2^{m-1}, k)$ is k. There is only one k remote-arc path.

The $k+1$ remote-arc path used in our algorithm means that we can add an additional hypercube cluster from the source hypercube cluster x to the destination hypercube cluster x'. It will go through $k+2$ hypercube clusters because the diameter of $K(2^{m-1}, k+1)$ is $k+1$. There are $(2^{m-1}-1)$ numbers of $k+1$ remote-arc paths. Now, we give details regarding how to select the next node in $KC(m,k)$. Any node in $KC(m,k)$ is labelled $\langle x = x_k...x_2x_1, y = y_m...y_2y_1\rangle$.

Firstly, we use $\langle x = x_k...x_2x_1, y = y_m...y_2y_1\rangle$ as a source node x and $\langle x' = x_k^{'}...x_2^{'}x_1^{'}, y^{'} = y_m^{'}...y_2^{'}y_1^{'}\rangle$ as a destination node x' in $KC(m,k)$.

Secondly, we mark the second digit of Kautz-part-label in the source node to the first digit of the Kautz-part-label in the destination node by a clockwise direction in a ring consisting of the values 0,1,..., d in ascending order if $x_1 \neq x_k^{'}$. If $x_1 = x_k^{'}$, we directly choose the k remote-arc path. There are three major paths that we need to consider. The first path is that the source node $\langle x = x_k...x_2x_1, y = y_m...y_2y_1\rangle$ routes to the next hypercube cluster $\langle x = x_{k-1}...x_1x_k^{'}, y = \bar{y_m}...\bar{y_2}y_1\rangle$. This is determined by the self-routing of Kautz digraph [31] on the k remote-arc path.

According to the construction rule of $KC(m,k)$, we can derive the i^{th} order of the out-arc from the source hypercube cluster $\langle x = x_k...x_2x_1, y = y_m...y_2y_1\rangle$ to the first intermediate hypercube cluster $\langle x = x_{k-1}...x_1x_k^{'}, y = y_{\bar{m}}...y_{\bar{2}}y_1\rangle$ and continue to reach the destination hypercube cluster $\langle x = x_k^{'}...x_2^{'}x_1^{'}, y = y_m^{'}...y_2^{'}y_1^{'}\rangle$ by the i^{th} order.

The second path is the source hypercube cluster $\langle x = x_k...x_2x_1, y = y_m...y_2y_1\rangle$ routes to the first intermediate hypercube cluster $\langle x = x_{k-1}...x_1\beta, y = y_{\bar{m}}...y_{\bar{2}}y_1\rangle$, $\beta \in \{0,1,...,d\} - \{x_1\}$. Here, i^{th} order indicates the clockwise distance from x_{k-1} to $x_k^{'}$ on the ring. They have the same i^{th} orders along intermediate hypercube clusters except the source hypercube cluster and destination hypercube cluster.

The third path is the source node $\langle x = x_k...x_2x_1, y = y_m...y_2y_1\rangle$ routes to the next hypercube cluster $\langle x = x_{k-1}...x_1\gamma, y = y_{\bar{m}}...y_{\bar{2}}y_1\rangle$, $\gamma \in \{0,1,...,d\} - \{x_1\}$. Here, i^{th} order indicates the clockwise distance from $x_{k-1}^{'}$ to x_k on the ring. They have the same i^{th} orders along intermediate hypercube clusters except the source hypercube cluster and destination hypercube cluster.

Thirdly, among the above three major middle paths, we could pick the best one, i.e., the one that has the fewest gaps between i^{th} remote-arc connections on the Kautz-part-label in the KCube.

Lastly, after choosing one of the three major paths, we can also check that the source hypercube cluster and the destination hypercube cluster on the hypercube-part-labels have the shortest hypercube routing paths. According to above analysis, we can derive the order j of the out-arc from node x to node v among all out-arcs of node x. Then the message should be forwarded from the j^{th} output node in hypercube cluster x to the j^{th} input node in the hypercube cluster v. If the node $\langle x, y\rangle$ is not the

j^{th} output node in the hypercube x, it infers the label of the j^{th} output node in the same hypercube, and routes the message to the j^{th} output node using the self-routing of hypercube [35]. In conclusion, we can have shorter paths for any pairs of nodes in $KC(m, k)$.

An example is shown in Figure 3.4. The source hypercube cluster is $\langle 1432, 0000 \rangle$, and the destination hypercube cluster is $\langle 0412, 1010 \rangle$ in $KC(4, 4)$. We mark the second digit of the Kautz-part-label in the source hypercube cluster to the first digit of the Kautz-part-label in the destination hypercube cluster by moving in a clockwise direction on a ring consisting of the values 0, 1, 2, 3,..., 8 in ascending order. In addition, since $x_1 \neq x_k^{'}$, we need to choose an additional hypercube cluster. According to the above method, we choose one of three major paths on the Kautz level connection. The first path is $\langle 1432 \rangle \rightarrow \langle 4320 \rangle \rightarrow \langle 3204 \rangle \rightarrow \langle 2041 \rangle \rightarrow \langle 0412 \rangle$ on the k remote-arc path. The middle intermediate hypercube clusters' i^{th} orders are $(7 - 1) = 6$ and $(7 - 1) = 6$.

The second path is on the $k + 1$ remote-arc path. The i^{th} order indicates the clockwise distance from x_{k-1} to $x_k^{'}$ on the ring. Each of these paths has the same i^{th} orders along intermediate hypercubes clusters except the source hypercube cluster and destination hypercube cluster. The middle intermediate hypercube cluster' i^{th} orders are $(6 - 2) = 4$ and $(8 - 2) = 6$. One of the paths is $\langle 1432 \rangle \rightarrow \langle 4324 \rangle \rightarrow \langle 3240 \rangle \rightarrow \langle 2404 \rangle \rightarrow \langle 4041 \rangle \rightarrow \langle 0412 \rangle$.

The third path is on the $k+1$ remote-arc path. The i^{th} order indicates the clockwise distance from $x_{k-1}^{'}$ to x_k on the ring. They have the same i^{th} orders along intermediate hypercube clusters except the source hypercube cluster and destination hypercube

cluster. The middle intermediate hypercube cluster' i^{th} orders are $(5-2)=3$ and $(8-2)=6$. One of the paths is $\langle 1432\rangle \rightarrow \langle 4323\rangle \rightarrow \langle 3230\rangle \rightarrow \langle 2304\rangle \rightarrow \langle 3041\rangle \rightarrow \langle 0412\rangle$. We choose the third one because it has the fewest gaps between i^{th} remote-arcs on the Kautz-part-label. After choosing the third major path, we will analyze the source hypercube cluster and the destination hypercube cluster's hypercube-part-labels to find the shortest routing paths. The shortest routing paths are usually to be considered the closest to the output node of the source node's hypercube-part-label and the closest input node of the destination node's hypercube-part-label. Because we have a ring, we don't need to figure out all the routing paths between the source node and the destination node. We use constant time to consider the three major paths on a ring and pick the one with the fewest gaps between i^{th} remote-arcs on the Kautz level connections. We also use constant time to find the smallest routing paths only in the source hypercube cluster and the destination hypercube cluster. This analysis takes constant time to compute. Since the first j^{th} order has been determined, we can derive the order j of the out-arc from node x to node v among all out-arcs of node x. Then the message should be forwarded from the j^{th} output node in hypercube x to the j^{th} input node in the hypercube v and continue to the destination node. It takes constant time for each node to decide what to do next. So the total running time required for routing on the KCube is proportional to the length of the routing paths, which is $O(km)$. The 6th path is the shortest path from the source node $< 1432, 0000 >$ to the destination node $< 0412, 1010 >$. We need to use constant time to analyze the j^{th} (6th) order among the three major paths. Then, we can pick the best one.

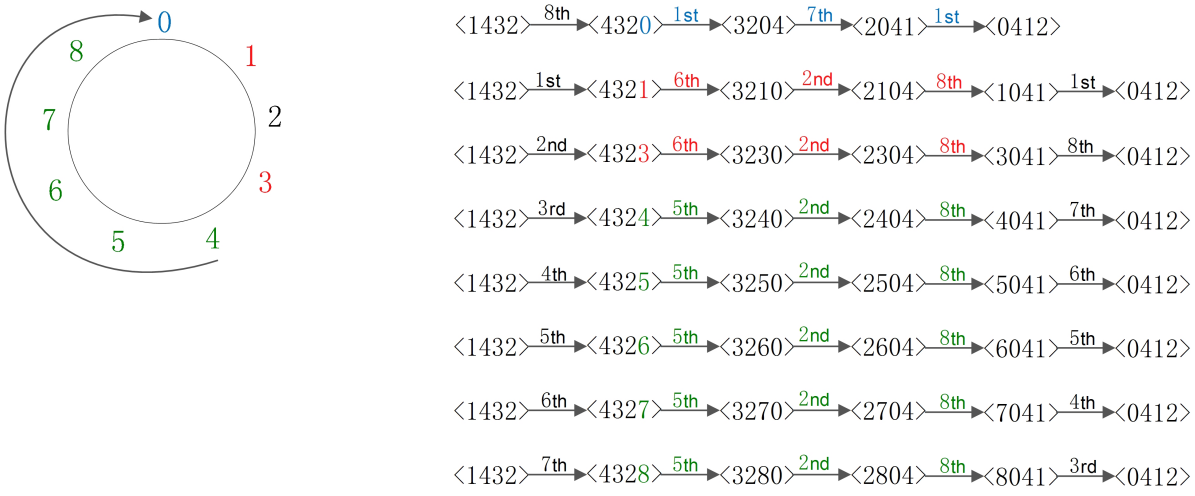


Figure 3.4: i^{th} Orders on the Kautz Levels of KC(4,4)

Lemma 2. *An upper bound on the diameter of the* $KC(m,k)$ *is* $m \times k + 2$.

Proof. The proof of an upper bound on the diameter of the $KC(m,k)$ from Guo et al. is $2m + (k-1)(m-1) + k = m(k+1) + 1$ [20]. However, we think this upper bound is not tight. Because we directly confirmed that the exact diameter of $KC(2,2)$ is 6, while Guo et al.'s upper bound is $2(2+1)+1 = 7$. We claim that an upper bound on the diameter of the $KC(m,k)$ is $m \times k + 2$. When we have two arbitrary nodes $\langle x, y\rangle$ and $\langle x', y'\rangle$, we use the routing Algorithm 1 to decide which routing path is selected. For any pairs of nodes, there are only two cases that we consider that they are either on the the k remote-arc path or $k+1$ remote-arc paths. The largest length of a shortest path from two arbitrary nodes $\langle x, y\rangle$ and $\langle x', y'\rangle$ on the k remote-arc path is guaranteed by the diameter of $K(2^{m-1}, k)$, and there are $k+1$ hypercube clusters involved. In the source hypercube x, the largest length of a shortest path from the node $\langle x, y\rangle$ to any other node is less than or equal to m, and is only equal to m when the node $\langle x, y\rangle$ is an output node and the other node is

$\langle x, \bar{y} \rangle$. The destination hypercube cluster x' receives a message from one of its input node and forwards to the node $\langle x', y' \rangle$ in one step. All the intermediate hypercube clusters traverse a maximum of $m-1$. There are $k-1$ such intermediate hypercube clusters. Also, the shortest path traverses k remote links that each link connects a pair of hypercubes clusters. Thus, an upper bound on the diameter of $KC(m,k)$ is $m+1+k+(k-1)(m-1) = km+2$. □

We generalize the routing path between any two nodes $\langle x, y \rangle$ and $\langle x', y' \rangle$ on the k remote-arc path as follows:

We use $\langle x = x_k...x_i...x_2x_1, y = y_m...y_i...y_2y_1 \rangle$ as a reference node, where $x_i \in \{0, 1, ..., 2^{m-1}\}$ and $1 \leq i \leq k$ and $y_i = 0$ or 1 for all $1 \leq i \leq m$.

$\langle x, y \rangle = \langle x_k...x_i...x_2x_1, y_m...y_i...y_2y_1 \rangle$

...

$\langle x_k...x_i...x_2x_1, \bar{y_m}...\bar{y_i}...\bar{y_2}\bar{y_1} \rangle$

...

$\langle x_{k-1}...x_i...x_10, \bar{y_m}...\bar{y_i}...\bar{y_2}y_1 \rangle$

...

$\langle x_{k-1}...x_i...x_10, y_m...y_i...y_2y_1 \rangle$

...

$\langle x_ix_{i-1}...x_10x_{k-1}x_{k-2}x_{i+1}, y_m...y_i...y_2\bar{y_1} \rangle$

...

$\langle x_ix_{i-1}...x_10x_{k-1}x_{k-2}x_{i+1}, \bar{y_m}...\bar{y_i}...\bar{y_2}\bar{y_1} \rangle$

...

$\langle x_{i-1}...x_1 0 x_{k-1}x_{k-2}x_{i+1}x_{i-2}, \bar{y_m}...\bar{y_i}...\bar{y_2}y_1\rangle$

...

$\langle x_{i-1}...x_1 0 x_{k-1}x_{k-2}x_{i+1}x_{i-2}, y_m...y_i...y_2y_1\rangle$

...

$\langle x_2x_1 0 x_{k-1}x_{k-2}x_{i+1}x_{i-2}...x_3, y_m...y_i...y_2\bar{y_1}\rangle$

...

$\langle x_2x_1 0 x_{k-1}x_{k-2}x_{i+1}x_{i-2}...x_3, \bar{y_m}...\bar{y_i}...\bar{y_2}\bar{y_1}\rangle$

...

$\langle x_1 0 x_{k-1}x_{k-2}x_{i+1}x_{i-2}...x_4, \bar{y_m}...\bar{y_i}...\bar{y_2}y_1\rangle$

...

$\langle x_1 0 x_{k-1}x_{k-2}x_{i+1}x_{i-2}...x_4, y_m...y_i...y_2y_1\rangle$

...

$\langle 0 x_{k-1}x_{k-2}x_{i+1}x_{i-2}...x_1, y_m...y_i...y_2\bar{y_1}\rangle$

...

$\langle 0 x_{k-1}x_{k-2}x_{i+1}x_{i-2}...x_1, \bar{y_m}...\bar{y_i}...\bar{y_2}\bar{y_1}\rangle = \langle x', y'\rangle$

We generalize the routing path between any two nodes $\langle x, y\rangle$ and $\langle x', y'\rangle$ on the $k+1$ remote-arc path below:

$\langle x, y\rangle = \langle x_k...x_i...x_1, y_m...y_i...y_2\bar{y_1}\rangle$

...

$\langle x_k...x_i...x_1, y_m...y_i...\bar{y_2}\bar{y_1}\rangle$

...

$\langle x_{k-1}...x_1x_i, y_m...y_i...\bar{y_2}y_1\rangle$ $i \neq k$, otherwise $i+1$.

...

$\langle x_{k-1}...x_1x_i, y_m...y_i...\bar{y_2}\bar{y_1}\rangle$

...

it repeats the same steps as above from i^{th} node to the destination node $\langle x', y'\rangle$.

...

$\langle 0x_{k-1}x_{k-2}x_{i+1}x_{i-2}...x_1, \bar{y_m}...\bar{y_i}...\bar{y_2}\bar{y_1}\rangle = \langle x', y'\rangle$ on $k+1$ remote-arc path.

On the $k+1$ remote-arc path, there are $k+2$ hypercube clusters. The source hypercube cluster sends a message directly to the next hypercube cluster without routing inside. The destination hypercube cluster receives the message in one hop. All the intermediate hypercube clusters traverse a maximum of $m-1$ steps. There are k intermediate hypercube clusters and $k+1$ remote arcs. Thus, it is $k(m-1)+1+k+1 = km+2$.

We claim that $km+2$ is a lower bound on the $KC(m,k)$ when $m=2$ or $m=3$ because any routing path that traverses larger than $k+1$ remote-arc path is already greater than $km+2$. Because the lower bound and upper bound are exactly same, we say the diameter of the $KC(m,k)$ is $m \times k+2$ when $m=2$ or $m=3$. Here, we give several examples to show how the routing algorithm works.

For example, we consider the routing path from $\langle 21, 11\rangle$ to $\langle 20, 01\rangle$ in $KC(2,2)$. We choose the path $\langle 21, 11\rangle \rightarrow \langle 10, 10\rangle \rightarrow \langle 10, 11\rangle \rightarrow \langle 02, 10\rangle \rightarrow \langle 02, 00\rangle \rightarrow \langle 20, 01\rangle$ on the $k+1$ remote-arc path, rather than the one Guo et al. chose on the k remote-arc path. They chose the path $\langle 21, 11\rangle \rightarrow \langle 21, 10\rangle \rightarrow \langle 21, 00\rangle \rightarrow \langle 12, 01\rangle \rightarrow \langle 12, 11\rangle \rightarrow \langle 20, 10\rangle \rightarrow \langle 20, 00\rangle \rightarrow \langle 20, 01\rangle$.

If the routing is from $\langle 21, 11\rangle$ to $\langle 20, 10\rangle$, we choose the path $\langle 21, 11\rangle \rightarrow \langle 21, 10\rangle \rightarrow \langle 21, 00\rangle \rightarrow \langle 12, 01\rangle \rightarrow \langle 12, 11\rangle \rightarrow \langle 20, 10\rangle$ on the k path rather than the path

$\langle 21,11\rangle \rightarrow \langle 10,10\rangle \rightarrow \langle 10,11\rangle \rightarrow \langle 02,10\rangle \rightarrow \langle 02,00\rangle \rightarrow \langle 20,01\rangle \rightarrow \langle 20,00\rangle \rightarrow \langle 20,10\rangle$ on the $k+1$ path.

It is possible for two nodes have the same routing distance on the k remote-arc path and the $k+1$ remote-arc path on the Kautz level.

If the path is from $\langle 21,11\rangle$ to $\langle 20,11\rangle$, the path $\langle 21,11\rangle \rightarrow \langle 10,10\rangle \rightarrow \langle 10,11\rangle \rightarrow \langle 02,10\rangle \rightarrow \langle 02,00\rangle \rightarrow \langle 20,01\rangle \rightarrow \langle 20,11\rangle$ is on the $k+1$ remote-arc path.

The path $\langle 21,11\rangle \rightarrow \langle 21,10\rangle \rightarrow \langle 21,00\rangle \rightarrow \langle 12,01\rangle \rightarrow \langle 12,11\rangle \rightarrow \langle 20,10\rangle \rightarrow \langle 20,11\rangle$ is on the k remote arc path. These two are the same. When a path in $KC(m,k)$ is identical to the k remote-arc path and $k+1$ remote-arc path, it is the exact diameter of the $KC(m,k)$.

Now consider routing in $KC(3,3)$ from $\langle 123,000\rangle$ to $\langle 021,111\rangle$, the first path is $\langle 123,000\rangle \rightarrow \langle 123,100\rangle \rightarrow \langle 234,101\rangle \rightarrow \langle 234,100\rangle \rightarrow \langle 340,101\rangle \rightarrow \langle 340,111\rangle \rightarrow \langle 402,110\rangle \rightarrow \langle 402,100\rangle \rightarrow \langle 021,101\rangle \rightarrow \langle 021,111\rangle$ on the $k+1$ remote-arc path. This path contains nine hops.

The second path is $\langle 123,000\rangle \rightarrow \langle 231,001\rangle \rightarrow \langle 231,011\rangle \rightarrow \langle 231,111\rangle \rightarrow \langle 310,110\rangle \rightarrow \langle 310,111\rangle \rightarrow \langle 102,110\rangle \rightarrow \langle 102,100\rangle \rightarrow \langle 102,000\rangle \rightarrow \langle 021,001\rangle \rightarrow \langle 021,011\rangle \rightarrow \langle 021,111\rangle$ on the $k+1$ remote-arc path. This path has eleven hops.

The third path is $\langle 123,000\rangle \rightarrow \langle 123,001\rangle \rightarrow \langle 123,011\rangle \rightarrow \langle 123,111\rangle \rightarrow \langle 230,110\rangle \rightarrow \langle 230,010\rangle \rightarrow \langle 230,000\rangle \rightarrow \langle 302,001\rangle \rightarrow \langle 302,011\rangle \rightarrow \langle 302,111\rangle \rightarrow \langle 021,110\rangle \rightarrow \langle 021,111\rangle$ on the k remote-arc path. This path contains eleven hops.

The forth path is $\langle 123,000\rangle \rightarrow \langle 123,001\rangle \rightarrow \langle 123,011\rangle \rightarrow \langle 232,010\rangle \rightarrow \langle 232,000\rangle \rightarrow \langle 232,100\rangle \rightarrow \langle 320,101\rangle \rightarrow \langle 320,111\rangle \rightarrow \langle 202,110\rangle \rightarrow \langle 202,111\rangle \rightarrow \langle 021,110\rangle \rightarrow \langle 021,111\rangle$ on the k remote-arc path. This path contains eleven hops.

We consider routing in $KC(3,3)$ from $\langle 123,000\rangle$ to $\langle 021,010\rangle$.

The path is $\langle 123,000\rangle \rightarrow \langle 123,001\rangle \rightarrow \langle 123,011\rangle \rightarrow \langle 123,111\rangle \rightarrow \langle 230,110\rangle \rightarrow \langle 230,010\rangle \rightarrow \langle 230,000\rangle \rightarrow \langle 302,001\rangle \rightarrow \langle 302,011\rangle \rightarrow \langle 302,111\rangle \rightarrow \langle 021,110\rangle \rightarrow \langle 021,010\rangle$ on the k remote-arc path. This path contains eleven hops.

The path is $\langle 123,000\rangle \rightarrow \langle 231,001\rangle \rightarrow \langle 231,011\rangle \rightarrow \langle 231,111\rangle \rightarrow \langle 310,110\rangle \rightarrow \langle 310,111\rangle \rightarrow \langle 102,110\rangle \rightarrow \langle 102,100\rangle \rightarrow \langle 102,000\rangle \rightarrow \langle 021,001\rangle \rightarrow \langle 021,000\rangle \rightarrow \langle 021,010\rangle$ on the $k+1$ remote-arc path. This path contains eleven hops.

Chapter 4

Broadcasting

4.1 Introduction

A broadcasting problem is an information dissemination process in a network. In a network, one processor sends a piece of information to all other processors. This process is to be completed as quickly as possible subject to a constraint. The constraint is that, in a single-port (weak) model, a processor is only allowed to send data to, or receive data from, one of its neighbours in one unit of time. Since after each unit of time, the number of processors with the data can at most be doubled, the broadcasting problem (BP) in such a model has a lower bound $\Omega(\log N)$ where N is the number of processors in the network. For the KCube network $KC(m, k)$, the lower bound will be $\Omega(\log((d^k + d^{k-1}) \times 2^m)) = \Omega(\log((d^k + d^{k-1}) \times 2d)) = \Omega(k \log d)$. The idea to design a broadcasting algorithm on the KCube is first to convert the KCube into several bipartite graphs. Because these bipartite graphs have the exact same structures, we can design a broadcasting algorithm to send data on these

bipartite graphs. Since the KCube is a compound graph of a Kautz digraph and hypercubes, the way to convert a KCube into several bipartite graphs is to partition the corresponding Kautz digraph into several complete bipartite graphs. The nodes of the partitioned Kautz complete bipartite graphs can be used as Kautz-part-labels in the KCube, and we can add hypercube nodes in each complete bipartite graph, which will result in bipartite graphs of the KCube. However, a Kautz digraph is a subdigraph of the corresponding de Bruijn digraph, so we need to understand how a de Bruijn digraph can be partitioned into complete bipartite graphs. The $K(d, k)$ is a subdigraph of the corresponding de Bruijn digraph. The corresponding de Bruijn digraph is $B(d+1, k)$. In fact, it is not an original de Bruijn digraph that can be partitioned into complete bipartite graphs, but actually the corresponding line graph of the de Bruijn digraph that can be partitioned into complete bipartite graphs. Many researchers have mentioned using line graphs because line graphs have good properties. The de Bruijn and Kautz complete bipartite graphs and their corresponding line graphs are isomorphic to each other. The relation between an original digraph G and its line graph $L(G)$ is that the $L(G)$ is isomorphic to the subsequent graph of G which is from the same d set of the original graph G (where d is the degree of each vertex). To design a broadcasting algorithm on the de Bruijn graph, we need to partition the corresponding line graph of the de Bruijn graph into complete bipartite graphs. Heydemann and Sotteau gave a method for partitioning de Bruijn graphs into several complete bipartite graphs [23]. Once we have de Bruijn complete bipartite graphs, we can design an algorithm on one bipartite graph and it will apply equally to all bipartite graphs. When we take out the nodes which are labelled with consecutive

letters (e.g., 000, 111) in the de Bruijn complete bipartite graphs, the resulting graphs are Kautz complete bipartite graphs. Bermond and Perennes designed a broadcasting algorithm on complete bipartite graphs of de Bruijn graphs and Kautz graphs which will be described in Section 4.3 [4]. What we do in our research is to take the Kautz partitioned complete bipartite graphs, and replace each node in the Kautz graph with a hypercube $H(m)$ with 2^m nodes. Inside each KCube bipartite graph, we set the left part nodes as initial nodes and we set the right part nodes as terminal nodes. The initial nodes are output nodes in the KCube. The terminal nodes are input nodes in the KCube. After adding the hypercube nodes, each KCube bipartite graph has the exact same structure. Our research is to design a broadcasting algorithm based on Bermond and Perennes's idea that utilize the complete bipartite graphs of a Kautz digraph and accommodate the hypercube lables of the KCube. This allows us to define a bipartite protocol on a KCube bipartite graph which will apply for all KCube bipartite graphs.

In this chapter, we will first illustrate the topological properties of the line graph and the complete bipartite graph. Then, we will show that the de Bruijn digraph and Kautz digraph's corresponding line graphs can be partitioned into several complete bipartite graphs. Third, we will explore Bermond and Perennes's broadcasting algorithm on bipartite graphs of de Bruijn graphs and Kautz graphs. Finally, based on Bermond and Perennes's broadcasting algorithm [4], we will develop two optimal broadcasting algorithms on the KCube.

4.2 Properties of the Line Graph and the Complete Bipartite Graph

In this section, we will begin by providing notations. Then, we will show the relationships between the de Bruijn graph and the Kautz graph. Lastly, we will list the topological properties of the de Bruijn graph. It is significant to study these properties because if we know the de Bruijn bipartition, we automatically know the Kautz bipartition according to the relationships between those two. We can use these properties to design broadcasting algorithms on KCube bipartite graphs later.

Given a connected graph G and a message originator, vertex u, the broadcast time of u, denoted $b(u)$, is the minimum number of time units required to complete broadcasting from u. The broadcast time of the graph G, $b(G)$, is defined as the maximum of $b(u)$ taken over all the vertices u in G.

The Kautz digraph is a subdigraph of de Bruijn digraph induced by the set of vertices without two consecutive identical letters. Before we explore the KCube bipartite graphs, it is important to examine the de Bruijn and Kautz digraphs' line graphs and bipartite graphs, which serve as the foundation of the KCube. The following results and topological properties are from [4, 23].

It is well known that the de Bruijn digraph (as well as the Kautz digraph) of indegree and outdegree d and diameter k is the line digraph of the same digraph of diameter $k-1$. Here the line digraph and the complete bipartite digraphs include directions because the de Bruijn and the Kautz digraphs have directions. Later, we remove all the directions in the line digraph and the complete bipartite digraphs

because they are bi-directional. As usual, $K_{d,d}$ denotes the bipartite digraph formed by two independent sets A and B of d vertices, with all arcs from A to B. We will use $K^0_{d,d}$ to denote the digraph obtained from $K_{d,d}$ by identifying two given adjacent vertices and replacing the arc joining them by a loop. The alphabet of size d is $\Sigma = \{0, 1, ..., d-1\}$. As a short form, we will use a^k to denote the word of length k composed of k occurrences of the letter a.

As illustrated above in the relationship between the line digraph and the original digraph, here we show how the line digraph of the de Bruijn digraph can be partitioned into the complete bipartite digraphs.

The line graph of the $B(d, k-1)$ is regular of indegree and outdegree d, then, the vertices of $B(d, k)$ corresponding to the in-coming and out-going arcs of a vertex without loop of $B(d, k-1)$ form a complete bipartite digraph $K_{d,d}$ in $B(d, k)$. The vertices of $B(d, k)$ corresponding to the in-coming and out-going arcs of a vertex a^{k-1} (with a loop) of $B(d, k-1)$ form a complete digraph $K^0_{d,d}$ [23]. We can have the following properties.

4.2.1 The Topological Properties of de Bruijn Digraph

- Any vertex $x_1x_2...x_k$ of $B(d, k)$, different from a^k, $a^{k-1}\alpha$, αa^{k-1} ($0 \leq a \leq d-1, 0 \leq \alpha \leq d-1, \alpha \neq a$) belongs to exactly two $K_{d,d}$'s. One of the $K_{d,d}$'s has vertex set the union of $A = \{\alpha x_2...x_k \mid 0 \leq \alpha \leq d-1\}$ and $B = \{x_2...x_k\beta \mid 0 \leq \beta \leq d-1\}$ and contains all the arcs from A to B. The vertex set of the other $K_{d,d}$ is the union of $\{x_1x_2...x_{k-1}\beta \mid 0 \leq \beta \leq d-1\}$ and $\{\alpha x_1x_2...x_{k-1} \mid 0 \leq \alpha \leq d-1\}$.

The vertex $x_1x_2...x_k$ has indegree 0 and outdegree d in the first $K_{d,d}$ and outdegree 0 and indegree d in the other one [23].

- For any letter a of the alphabet ($0 \leq a \leq d-1$), the vertex a^k of $B(d,k)$ belongs to the $K^0_{d,d}$ with the vertex set the union of $A = \{\alpha a^{k-1} \mid 0 \leq \alpha \leq d-1, \alpha \neq a\}$, $B = \{a^{k-1}\beta \mid 0 \leq \beta \leq d-1, \beta \neq a\}$, and $\{a^k\}$. The subdigraph $K^0_{d,d}$ contains all the arcs from A to B, from A to a^k, from a^k to B and a loop on a^k, so a^k has both indegree and outdegree d in the $K^0_{d,d}$ [23].

- Vertices of $B(d,k)$ of the form $a^{k-1}\alpha$ (or αa^{k-1}), with $0 \leq a \leq d-1$, $0 \leq \alpha \leq d-1$, $\alpha \neq a$, belong to one subdigraph $K_{d,d}$ with the vertex set the union of $A = \{\gamma a^{k-2}\alpha \mid 0 \leq \gamma \leq d-1\}$ and $B = \{a^{k-2}\alpha\beta \mid 0 \leq \beta \leq d-1\}$ and one subdigraph $K^0_{d,d}$ with the vertex set the union of $A = \{\alpha a^{k-1} \mid 0 \leq \alpha \leq d-1, \alpha \neq a\}$, $B = \{a^{k-1}\beta \mid 0 \leq \beta \leq d-1, \beta \neq \alpha\}$, and a^k [23].

- Each arc of $B(d,k)$ belongs to a unique $K_{d,d}$ or $K^0_{d,d}$. The arcs of $B(d,k)$ can be partitioned into d^{k-1} complete bipartite graphs ($K_{d,d}$). Therefore, the digraph $B(d,k)$ is the arc disjoint union of exactly $d^{k-1} - d$ digraphs, each one being isomorphic to $K_{d,d}$, and of d digraphs, each one being isomorphic to $K^0_{d,d}$ [23].

We have provided an example to show an original digraph's corresponding line digraph. According to the line graph definition, we use a vertex to represent each edge in $B(2,2)$ in the subsequent digraph. Then we add together all of the edges in terms of de Bruijn digraph connections. This process results in a line digraph for $B(2,2)$ which is $B(2,3)$. $B(2,3)$ is isomorphic to the subsequent graph of $B(2,2)$

which is from the same d ($d = 2$) set of the original graph $B(2, 2)$ (where d is the degree of each vertex). It is $B(2, 3) = L(B(2, 2))$. The same property holds for de Bruijn and Kautz digraphs, (i.e.,) $B(d, k) = L(B(d, k-1))$ and $K(d, k) = L(K(d, k-1))$.

According to the above de Bruijn properties, we have included an example to show how the line digraph could be partitioned into several complete bipartite digraphs. Later, we will remove all the directions in the complete bipartite digraphs as a result of bi-directionality. Since $B(2, 3)$ is a line digraph of $B(2, 2)$, we partition the line digraph $B(2, 3)$ into several bipartite digraphs. Indeed, since $B(2, 2)$ is a regular digraph, with indegree and outdegree 2, the vertices of $B(2, 3)$ corresponding to the in-coming and out-going arcs of a vertex without loop of $B(2, 2)$ form a complete bipartite digraph $K_{2,2}$ in $B(2, 3)$. The vertices of $B(2, 3)$ corresponding to the in-coming and out-going arcs of a vertex a^2 (with a loop) of $B(2, 2)$ form a complete digraph $K^0_{2,2}$. Each arc of $B(2, 3)$ belongs to a unique $K_{2,2}$ or $K^0_{2,2}$ (see Figure 4.1 and Figure 4.2). These complete bipartite digraphs have directions. Since the first $K_{2,2}$ and the second $K_{2,2}$ are bidirectional, we can remove all the directions. The arcs of $B(2, 3)$ can be partitioned into $d^{k-1} = 2^{3-1} = 4$ complete bipartite graphs ($K_{2,2}$) in Figure 4.3. The graph $B(2, 3)$ is the arc disjoint union of $d^{k-1} - d = 2^{3-1} - 2 = 2$ cycles of length 4, denoted C_4 (each one being isomorphic to $K_{2,2}$), and of 2 triangles (each one being isomorphic to $K^0_{2,2}$) in Figure 4.5. We remove all the directions in de Bruijn subgraphs because complete bipartite graphs are bi-directional. It is easy to find which complete bipartite graphs are isomorphic to the undirected $UB(2, 3)$ (as defined in Section 2.2) as seen in Figure 4.4. The corresponding properties for the undirected de Bruijn graph can be easily deduced. In particular, when $d = 2$, the

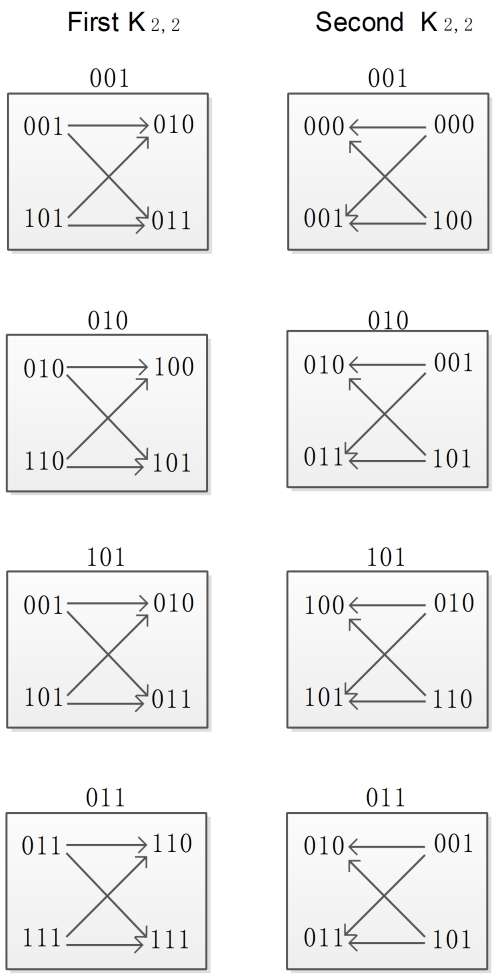


Figure 4.1: Partition $B(2,3)$ to $K_{2,2}$

undirected binary de Bruijn graph of diameter 3, denoted by $UB(2,3)$, is the union of 2 cycles of length 4, denoted C_4, and of two triangles, which are all edge disjoint in Figure 4.5.

4.3 A Broadcasting Algorithm on the Kautz Graph

In this section, we will show Bermond and Perennes's broadcasting algorithm on an undirected Kautz graph [4]. It is important to understand how the broadcasting algorithm executes on the Kautz graph because a KCube will partition the corresponding

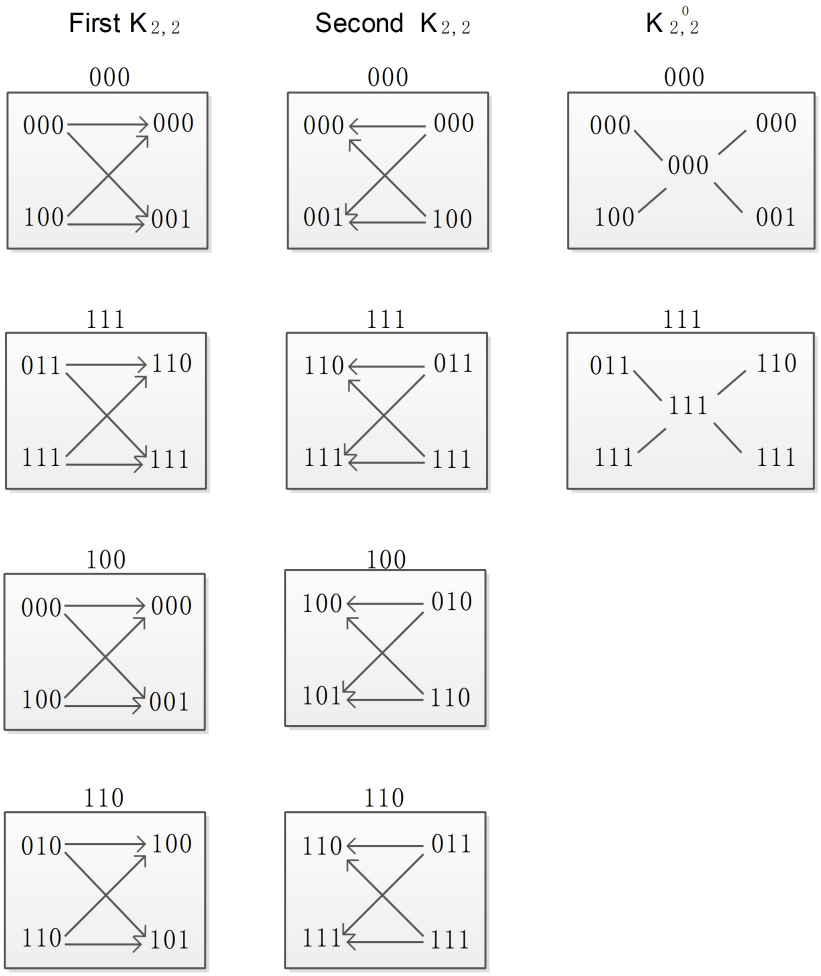


Figure 4.2: Partition $B(2,3)$ to $K_{2,2}$ and $K^0_{2,2}$

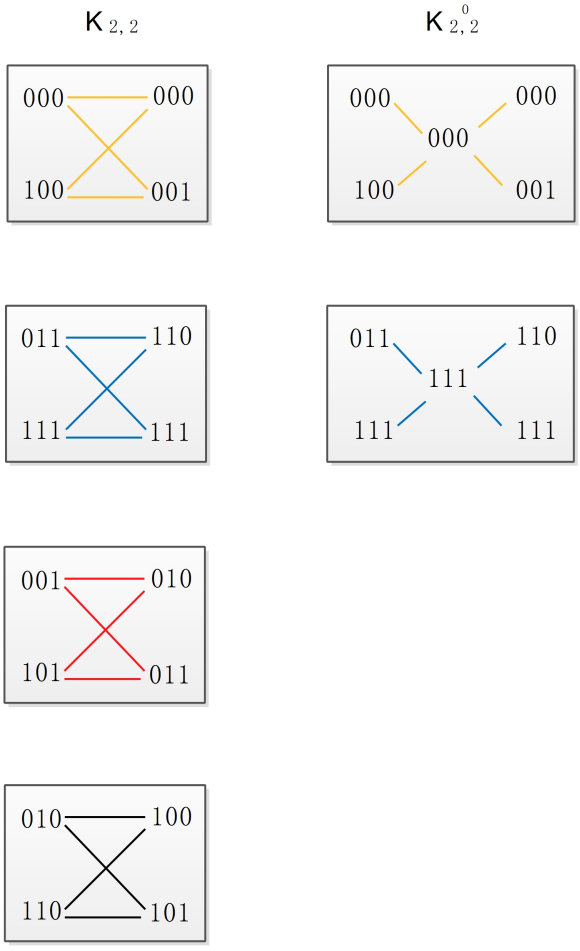


Figure 4.3: Undirected $K_{2,2}$ Graphs

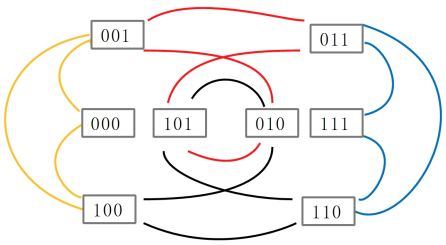


Figure 4.4: $UB(2,3)$

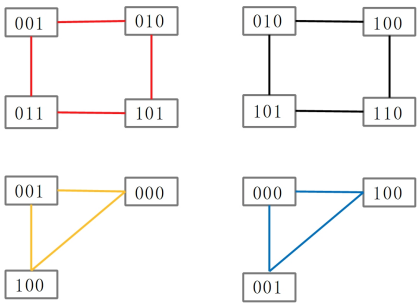


Figure 4.5: Four Disjoint Cycles

Kautz graph according to the Kautz bipartition and add all of the hypercube nodes on each complete bipartite graph. Later, we will design two optimal broadcasting algorithms on the KCube (Section 4.4).

$K(d,k)$ is the subdigraph of $B(d+1,k)$. $K(d,k)$ is generated using the set of vertices without two consecutive identical letters from $B(d+1,k)$. If a $K_{d+1,d+1}$ of $B(d+1,k)$ induces a $K_{d,d}$ in $K(d,k)$, similar properties hold for $K(d,k)$. In fact, the digraph $K(d,k)$ is the union of $d^{k-1}+d^{k-2}$ digraphs, each one isomorphic to the bipartite graphs $K_{d,d}$ [23].

A Bipartite Broadcasting Protocol

Bermond and Perennes explain how a bipartite protocol broadcasts on complete bipartite graphs (isomorphic to the line graph of the original Kautz graph). If they let the vertices of $K_{d,d}$ be, respectively, $A=\{a_0,...,a_{d-1}\}$ and $B=\{b_0,...,b_{d-1}\}$. The broadcasting protocol is as follows: at time 1, if the originator is a_i, it informs b_i; then at time $t \geq 2$, any vertex a_j (and ,respectively, b_j) which knows the message sends it to $b_{j+2^{t-2}}$ (and, respectively, $a_{j+2^{t-2}}$). By induction, one can easily show that after time t, if a_i is the originator, the message is known by 2^{t-1} vertices of B, namely $b_i,...,b_{i+2^{t-1}-1}$ and 2^{t-1} vertices of A, namely $a_i,...,a_{i+2^{t-1}-1}$ [4].

For example, consider the broadcasting algorithm running on the $K(2,2)$. $K(2,2)$ is the subdigraph of $B(3,2)$. This $B(3,2)$ digraph has some nodes with consecutive identical letters and can be partitioned into 3 complete bipartite graphs ($K_{3,3}$) in Figure 4.6. Because our work on the KCube is based on a Kautz digraph and the Kautz digraph doesn't have nodes with consecutive identical letters, we remove them (00, 11, 22) to deduce $K_{2,2}$ in $K(2,2)$ in Figure 4.7. The broadcasting procedure lets the vertices of the first $K_{2,2}$ be respectively $A = \{a_0, a_1\} = \{01, 21\}$ and $B = \{b_0, b_1\} = \{10, 12\}$. At time 1, a_0 informs b_0 which sends data from 01 to 10. At time $t \geq 2$, any vertex a_j (and, respectively, b_j) sends data to $b_{j+2^{t-2}}$ (and, respectively, $a_{j+2^{t-2}}$), that is a_0 sends data to b_1 and b_0 sends data to a_1. They are nodes 01 to 12 and 10 to 21. During each phase, each vertex, which has received the message as a terminal vertex of first $K_{2,2}$ in the preceding phase, sends data to the terminal vertices of the $K_{2,2}$, which are initial vertices in the other $K_{2,2}$ bipartite graphs. In this case, 01 and 10, 21 and 12, which are terminal vertices in the first $K_{2,2}$, have already been informed. But, they are also the initial vertices in the second $K_{2,2}$ and the third $K_{2,2}$. Each $K_{2,2}$ executes the bipartite protocol one more time. Then, the broadcasting is complete. Each $K_{2,2}$ takes $\log d$ time to send data. There are $d^{k-1} + d^{k-2} = 2^{2-1} + 2^{2-2} = 3$ complete bipartite graphs. $b(UK(2,2)) \leq k(\log d) = 3 \times 1 = 3$.

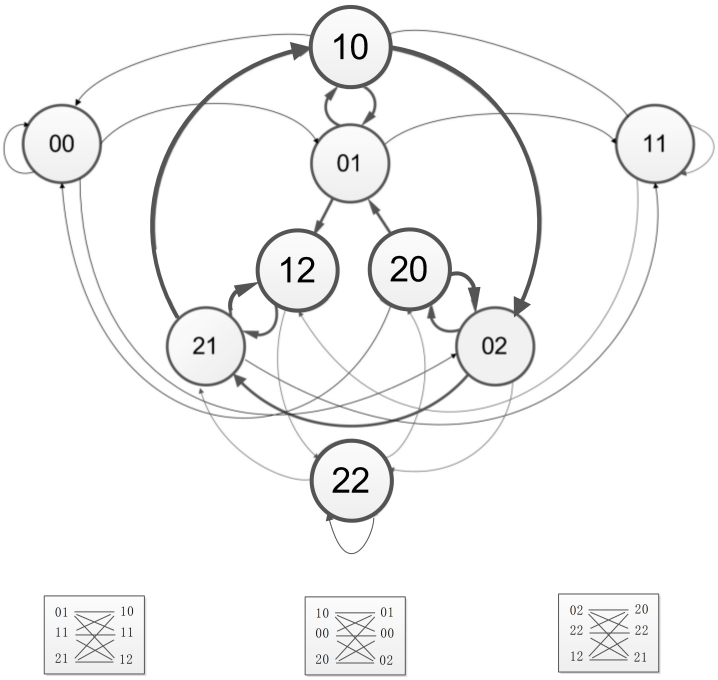


Figure 4.6: $B(3,2)$ and $K_{3,3}$

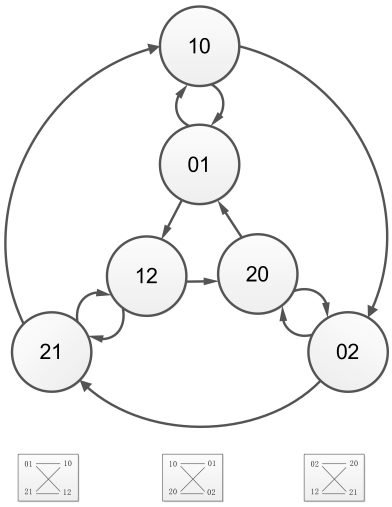


Figure 4.7: $K(2,2)$ and $K_{2,2}$

4.4 Two Optimal Broadcasting Algorithms on the KCube Graphs

The way to convert KCube into bipartite graphs is to partition the corresponding Kautz digraph as the Kautz-part-labels according to the Kautz bipartition [23] and add all the hypercube-part-labels on the KCube. The method for partitioning a Kautz digraph is presented in Section 4.3.

Each $K_{d,d}$ corresponds to a vertex of the KCube as follows. The $K_{d,d}$ is associated to $\langle x = x_k...x_2x_1, y = y_m...y_2y_1\rangle$ where

- $y_2y_1 = x_2x_1$ or $y_2y_1 = \bar{x_2}\bar{x_1}$
- $y_i = 0$ or 1 for all $3 \leq i \leq m$

Inside each KCube bipartite graph, we set the left part nodes as initial vertices (output nodes) and the right part nodes as terminal vertices (input nodes).

- $y_2y_1 = \bar{x_2}x_1$ or $y_2y_1 = x_2\bar{x_1}$
- $y_i = 0$ or 1 for all $3 \leq i \leq m$

of this $K_{d,d}$.

We are ready to broadcast in bipartite graphs of $KC(m,k)$ with a protocol that is the bipartite protocol for the $K_{d,d}$. The vertices of $K_{d,d}$ are respectively $A = \{a_0, ..., a_{d\times 2^{m-1}-1}\}$ and $B = \{b_0, ..., b_{d\times 2^{m-1}-1}\}$. The partition procedure proceeds in sequence from the first $K_{d,d}$ to the last $K_{d,d}$.

Partition Procedure:

1. $\langle x = x_k...x_2x_1, y = y_m...y_2y_1\rangle \rightarrow \langle x = x_{k-1}...x_2x_1\alpha, y = y_m...y_2\bar{y_1}\rangle$. $\langle x = x_k...x_2x_1, y = y_m...y_2y_1\rangle$ is an originator with a datum in the first KCube bipartite graph. It is an output node. We set this node is the first initial node denoted a_0 in the set A. $\langle x = x_{k-1}...x_2x_1\alpha, y = y_m...y_2\bar{y_1}\rangle$ is the first input node corresponding an originator in the first KCube bipartite graph. We set it as the first terminal node denoted b_0 in the set B.

2. This procedure the Kautz-part-label doesn't change, only differ from the rightmost one digit each time on the hypercube-part-label until differs all hypercube-part-label digits.

   $\langle x = x_k...x_2x_1, y = y_m...y_2y_1\rangle \rightarrow \langle x = x_k...x_2x_1, y = y_m...y_2\bar{y_1}\rangle$. We set $\langle x = x_k...x_2x_1, y = y_m...y_2\bar{y_1}\rangle$ as the second terminal node (input node) in set B. It denotes b_1. At the same time, $\langle x = x_{k-1}...x_2x_1\alpha, y = y_m...y_2\bar{y_1}\rangle \rightarrow \langle x = x_{k-1}...x_2x_1\alpha, y = y_m...\bar{y_2}\bar{y_1}\rangle$. We set $\langle x = x_{k-1}...x_2x_1\alpha, y = y_m...\bar{y_2}\bar{y_1}\rangle$ as the second initial node (an output node) denoted a_1 in the set A. Each partition procedure happens from an initial vertex to a terminal vertex and also from a terminal vertex to an initial vertex. This procedure ends until differs all the hypercube digits on the hypercube-part-label.

3. After differing all hypercube digits , Kautz-part-label starts to change according to the Kautz bipartite partition and repeat step 1. Each procedure happens from an initial vertex to a terminal vertex and also from a terminal vertex to an initial vertex. This procedure ends until changing each corresponding Kautz-part-label.

4. It starts to repeat the step 2 which doesn't change Kautz-part-labels and differ all hypercube-part-label digits.

5. It starts to repeat the step 3 which changes the Kautz-part-labels and doesn't change the hypercube-part-labels. This step finishes until the last KCube bipartite graph.

The first broadcasting algorithm is as follows: at time 1, if the originator is a_i, it informs b_i. At time $t \geq 2$, any vertex a_j (resp b_j) which knows the message sends it to $b_{j+2^{t-2}}$ resp $(a_{j+2^{t-2}})$. Then at time $t \geq 4$, any vertex a_p (resp b_p) which knows the message sends it to $b_{p+2^{t-3}}$ resp $(a_{p+2^{t-3}})$.

At time $t \geq 5$, any vertex a_p (resp b_p) which knows the message sends it to $b_{p+2^{t-4}}$ resp $(a_{p+2^{t-4}})$

By induction one can easily show that after time t, if a_p is the initial vertex, the message is known by 2^{t-4} vertices of B, namely, $b_p, ..., b_{p+2^{t-4}-1}$ and 2^{t-4} vertices of A, namely $a_p, ..., a_{p+2^{t-4}-1}$. In addition, we add the previous time nodes.

As an example, $KC(2,2)$ is a compound graph of the Kautz digraph $K(2,2)$ and Hypercubes $H(2)$. A constraint is $2d = 2^m \rightarrow d = 2^{m-1} = 2^{2-1} = 2$.
$K(2,2)$ is the subgraph of $B(3,2)$ induced by the set of vertices without two consecutive identical letters and a $K_{3,3}$ of $B(3,2)$ in Figure 4.6 induces a $K_{2,2}$ of $K(2,2)$ in Figure 4.7. In particular the $K(2,2)$ is the arc disjoint union of exactly $d^{k-1}+d^{k-2} = 3$ graphs, each one is isomorphic to the graph $K_{2,2}$. We can add hypercube-part-labels in Figure 4.8. We can partition all the nodes into two sets in the next table. Now, we are ready to broadcast a message according to the protocol. At time 1, $a_0 \longrightarrow b_0$.

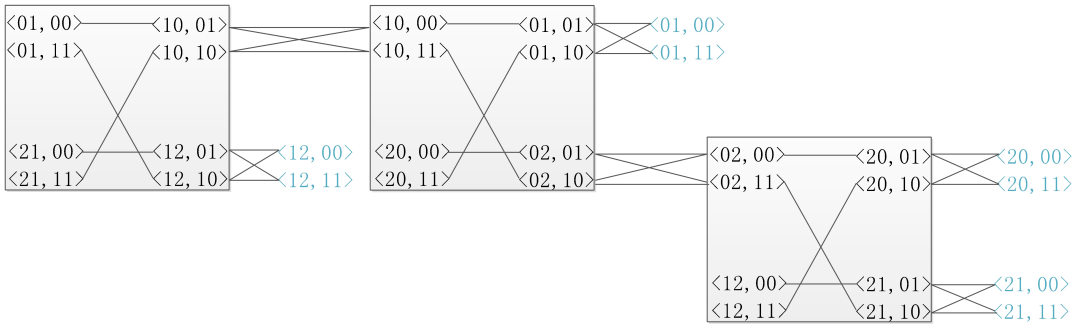


Figure 4.8: Partitions of $KC(2,2)$

At time 2, $a_0 \longrightarrow b_1$ and $b_0 \longrightarrow a_1$. At time 3, $a_0 \longrightarrow b_2$, $b_0 \longrightarrow a_2$, $b_1 \longrightarrow a_3$ and $a_1 \longrightarrow b_3$. At time 4, $b_2 \longrightarrow a_4$, $a_2 \longrightarrow b_4$, $a_3 \longrightarrow b_5$ and $b_3 \longrightarrow a_5$. At time 5, $a_4 \longrightarrow b_6$, $b_4 \longrightarrow a_6$, $b_5 \longrightarrow a_7$, and $a_5 \longrightarrow b_7$. At time 6, $a_4 \longrightarrow b_8$, $b_4 \longrightarrow a_8$, $b_5 \longrightarrow a_9$, $a_5 \longrightarrow b_9$, $b_6 \longrightarrow a_{10}$, $a_6 \longrightarrow b_{10}$, $a_7 \longrightarrow b_{11}$ and $b_7 \longrightarrow a_{11}$.

a_0	$< 01, 00 >$		$< 10, 01 >$	b_0
a_1	$< 10, 00 >$		$< 01, 01 >$	b_1
a_2	$< 10, 11 >$		$< 01, 10 >$	b_2
a_3	$< 01, 11 >$		$< 10, 10 >$	b_3
a_4	$< 20, 11 >$		$< 02, 10 >$	b_4
a_5	$< 21, 11 >$		$< 12, 10 >$	b_5
a_6	$< 02, 11 >$		$< 20, 10 >$	b_6
a_7	$< 12, 11 >$		$< 21, 10 >$	b_7
a_8	$< 02, 00 >$		$< 20, 01 >$	b_8
a_9	$< 12, 00 >$		$< 21, 01 >$	b_9
a_{10}	$< 20, 00 >$		$< 02, 01 >$	b_{10}
a_{11}	$< 21, 00 >$		$< 12, 01 >$	b_{11}

This protocol works with $KC(2,2)$, and it may work for arbitrary $KC(m,k)$. However, further research must be done to test this protocol. The difference between the first bipartite protocol and the second bipartite protocol is that there is an additional partition procedure in the first bipartite protocol. It partitions all nodes of KCube bipartite graphs into two sets A and B where we designed the protocol on the two sets. However, in the second bipartite protocol, there is no this partition procedure. We directly designed a bipartite protocol on remote arcs inside one KCube bipartite graph which can apply for all KCube bipartite graphs. It works for arbitrary $KC(m,k)$.

The second bipartite broadcasting protocol is to let the vertices of each $K_{d,d}$ be respectively $A = \{a_0, ..., a_{d\times 2^{m-1}-1}\}$ and $B = \{b_0, ..., b_{d\times 2^{m-1}-1}\}$. The broadcasting procedure has two phases. The first phase happens in the KCube bipartite graphs. We define a bipartite protocol on the KCube bipartite graphs. The second phase happens in the hypercube clusters with the same Kautz-part-labels. We use the hypercube broadcasting algorithm [6]. If we check the KCube bipartite graphs carefully, inside each KCube bipartite graph, the connections between the initial vertices (output nodes) and the terminal vertices (input nodes) are the remote arcs on the KCube. Recall that on the KCube, we call any arc which connects two nodes in different hypercube clusters a remote arc. We call any arc which connects two nodes in the same hypercube cluster a local arc. The advantage of representing a KCube as KCube bipartite graphs is that this approach gives us a way to simplify determining which nodes will send or receive data in which order. Each KCube bipartite graph is connected to other KCube bipartite graphs by local arcs, which gives us one

way to show how data can be transferred from one KCube bipartite graph to other KCube bipartite graphs, because after we convert the KCube into KCube bipartite graphs, we "split" each hypercube cluster into two equal parts. The "splitting" is virtual; the hypercube clusters are not split on the original KCube graph. One half of the hypercube cluster retains the same Kautz-part-label and all the hypercube-part-labels are output nodes. The other half retains the same Kautz-part-label and all the hypercube-part-labels are input nodes. For example, in Figure 4.8, in the first KCube bipartite graph, the two terminal nodes $< 10, 01 >$ and $< 10, 10 >$ are connected by local arcs to two initial nodes $< 10, 00 >$ and $< 10, 11 >$ in the next KCube bipartite graph. Every "split" hypercube cluster in a KCube bipartite graph is connected by local arcs to another "split" hypercube cluster in another KCube bipartite graph with the same Kautz-part-label. So, we can say on the KCube bipartite graphs, the remote arcs show how data transfers among different hypercube clusters and the local arcs show how data transfers inside each hypercube cluster.

So, the second broadcasting algorithm is as follows:

First, we use the first node (arbitrarily chosen) in the first KCube bipartite graph as an originator to send a datum along the first remote arc to the corresponding node $a_i \longrightarrow b_i$ in the same KCube bipartite graph. The two nodes (an initial node and a terminal node) that now have the data are called informed initial/terminal nodes.

Second, the two informed nodes, which are in different "split" hypercube clusters (in different KCube bipartite graphs) send data along the local arcs to the corresponding i^{th} input/output nodes in the other "split" hypercube clusters with the same Kautz-part-labels. For example, in Figure 4.8, at this step, the first output

node $< 01, 00 >$ sends a datum by the local arc to the corresponding first input node $< 01, 01 >$ in the other split hypercube cluster with the same Kautz-part-label. At the same time, the first input node $< 10, 01 >$ sends a datum by the local arc to the corresponding first output node $< 10, 00 >$ in the other "split" hypercube cluster with the same Kautz-part-label. In this example, $KC(2, 2)$, at step 2, 4 nodes are informed on the KCube. In Figure 4.8, the blue labels do not represent added nodes on the KCube bipartite graphs. Instead, they represent the labels of nodes which are connected by local arcs to nodes in a different KCube bipartite graph that is not contiguous in the diagram.

Third, in each informed "split" hypercube cluster, we use the hypercube broadcasting algorithm [6], which partitions $H(m)$ into 2 sub-hypercubes of size $H(m-1)$ and sends data from one sub-hypercube $H(m-1)$ to another sub-hypercube $H(m-1)$ in parallel by differing a significant bit. The advantage of the hypercube broadcasting algorithm is that both $H(m-1)$ hypercubes broadcast data independently in parallel. Broadcasting data on the $H(m)$ from one $H(m-1)$ to another $H(m-1)$ can be done in one unit of time, which allows us to double the number of processors with data inside each "split" hypercube cluster at each time. All the data transfers inside each informed "split" hypercube cluster in parallel along local arcs at the same time. This step allows all the nodes inside the informed hypercube clusters to get the data simultaneously using the hypercube broadcasting algorithm. For example, in Figure 4.8, the hypercube of $H(2)$ can partition into 2 sub-hypercubes of $H(1)$. The sub-hypercube of $H(1)$ (an edge with a node $< 10, 01 >$ and a node $< 10, 00 >$) can send data to another sub-hypercube of $H(1)$ (an edge with a node $< 10, 11 >$ and

a node $< 10, 10 >$) by differing the leftmost significant bit. At the same time, the sub-hypercube of $H(1)$ (an edge with a node $< 01, 01 >$ and a node $< 01, 00 >$) can send data to another sub-hypercube of $H(1)$ (an edge with a node $< 01, 11 >$ and a node $< 01, 10 >$) by differing the leftmost significant bit.

Fourth, the last nodes to be informed in step 3 send data to the corresponding nodes by the remote arcs inside each KCube bipartite graph with different Kautz-part-labels. Some of these nodes are initial nodes and some of these nodes are terminal nodes. Any vertex a_j (resp b_j) which has the datum sends it to b_{i+d+1} resp (a_{i+d+1}). The i subscript is the previous bipartite protocol defined on the remote arc at sept 1. The d is an indegree/outdegree of a Kautz graph. These remote arcs inside each KCube bipartite graphs give us a clear way to double the number of new informed hypercube clusters. For example, in Figure 4.8, an initial node a_1 ($< 01, 11 >$) can send a datum to a terminal node b_3 ($< 12, 10 >$) and an terminal node b_1 ($< 10, 10 >$) can send a datum to an initial node a_3 ($< 21, 11 >$) with different Kautz-part-labels using the same bipartite protocol in the first KCube bipartite graph. At the same time, an initial node a_1 ($< 10, 11 >$) can send a datum to a terminal node b_3 ($< 02, 10 >$) and a terminal node b_1 ($< 01, 10 >$) can send a datum to an initial node a_3 ($< 20, 11 >$) with different Kautz-part-labels using the same bipartite protocol in the second KCube bipartite graph.

Fifth, the informed nodes from step 4 send the data to the corresponding i^{th} input/output nodes in their own "split" hypercube clusters with the same Kautz-part-label. This step repeats step 2. For example, in Figure 4.8, at this step, the second input node $< 12, 10 >$ sends a datum by the local arc to the corresponding

second output node $< 12, 11 >$ in the other "split" hypercube cluster with the same Kautz-part-label. The second input node $< 02, 10 >$ sends a datum by the local arc to the corresponding second output node $< 02, 11 >$ in the other "split" hypercube cluster with the same Kautz-part-label. At the same time, the second output node $< 21, 11 >$ sends a datum by the local arc to the corresponding second input node $< 21, 10 >$ in the other "split" hypercube cluster with the same Kautz-part-label. The second output node $< 20, 11 >$ sends a datum by the local arc to the corresponding second input node $< 20, 10 >$ in the other "split" hypercube cluster with the same Kautz-part-label. At this step, 8 nodes are informed on the KCube.

Sixth, we repeat step 3 using the hypercube broadcasting algorithm, which sends data from $H(m-1)$ to the corresponding $H(m-1)$ by differing a significant bit. For example, in Figure 4.8, the hypercube of $H(2)$ can partition into 2 sub-hypercubes of $H(1)$. The sub-hypercube of $H(1)$ (an edge with a node $< 12, 10 >$ and a node $< 12, 11 >$) can send data to another sub-hypercube of $H(1)$ (an edge with a node $< 12, 00 >$ and a node $< 12, 01 >$) by differing the leftmost significant bit. The sub-hypercube of $H(1)$ (an edge with a node $< 02, 10 >$ and a node $< 02, 11 >$) can send data to another sub-hypercube of $H(1)$ (an edge with a node $< 02, 00 >$ and a node $< 02, 01 >$) by differing the leftmost significant bit. At the same time, the sub-hypercube of $H(1)$ (an edge with a node $< 21, 11 >$ and a node $< 21, 10 >$) can send data to another sub-hypercube of $H(1)$ (an edge with a node $< 21, 01 >$ and a node $< 21, 00 >$) by differing the leftmost significant bit. The sub-hypercube of $H(1)$ (an edge with a node $< 20, 11 >$ and a node $< 20, 10 >$) can send data to another sub-hypercube of $H(1)$ (an edge with a node $< 20, 01 >$ and a node

$< 20, 00 >$) by differing the leftmost significant bit. This step allowing different "split" hypercube clusters with different Kautz-part-labels in one KCube bipartite graph can send data to another "split" hypercube clusters in different KCube bipartite graphs simultaneously at one step in one KCube bipartite graph.

Seventh, if we still have some hypercube clusters left which are not informed yet, we can repeat step one to send data along the remote arcs and inform new hypercube clusters with different Kautz-part-labels.

Eighth, we can repeat step 2.

Ninth, we execute the hypercube broadcasting algorithm again.

Tenth, the procedure continues until all nodes been informed.

It is known that if G is a $d-regular$ digraph: $b(UL(G)) \leq D(L(G))(\log_2(d)+1)$ [4].

Time Analysis:

First of all, we could easily come up with a simple but inefficient broadcasting algorithm on the KCube. We could simply combine the Kautz broadcasting algorithm and the hypercube broadcasting algorithm. For example, on the KCube, we could apply the Kautz broadcasting algorithm first to broadcast data to hypercube cluster(s). Inside each hypercube cluster, we apply the hypercube broadcasting algorithm. After the hypercube broadcasting algorithm is finished, we could continue with the Kautz broadcasting algorithm to send data to the next hypercube cluster, and apply the hypercube broadcasting algorithm again, and so on. The optimal running time for the Kautz broadcasting algorithm is $\Omega(k \log d)$ because there are $d^k + d^{k-1}$ number of nodes in the Kautz graph. When we apply the Kautz broadcasting algorithm on the

KCube, however, each node in the Kautz graph is replaced with a hypercube cluster $H(m)$. Therefore, each node at each step in the original Kautz broadcasting algorithm will have to spend an extra $O(m)$ time to apply the hypercube broadcasting algorithm. Therefore, the total running time for broadcasting data in this way on the KCube is the Kautz broadcasting running time times the hypercube broadcasting running time, which is $O(m \times k \log d)$. However, it is not optimal since we know that the lower bound of the broadcasting algorithm on the KCube is $\Omega(k \log d)$.

What we have done is develop a bipartite broadcasting algorithm on the KCube. We first convert the KCube into a number of KCube bipartite graphs. Then, we design a bipartite protocol on the KCube bipartite graphs which allows us to broadcast data on them. The KCube bipartite graphs and the original KCube graph are isomorphic to each other. Therefore, if we are able to apply our broadcasting algorithm to the KCube bipartite graphs, we can broadcast data on the original KCube graph. Let $t(m)$ be the time that it takes to broadcast data in a hypercube cluster. Then, using the procedure we described above in the hypercube cluster, we send the data from one "split" hypercube cluster to the corresponding i^{th} input/output nodes with the same Kautz-part-label in another "split" hypercube cluster by using the hypercube broadcasting algorithm, which takes $\log d + 1$ time in a hypercube cluster. Because $t(m) = 1+t(m-1)$ and $m = \log N$, the N is the total number of nodes in $H(m)$, which is $N = 2^m$. The constraint in the KCube is $2d = 2^m \longrightarrow d = 2^{m-1} \longrightarrow m = \log d + 1$. According to the the Kautz partitions, each KCube bipartite graph contains a number of "split" hypercube clusters with different Kautz-part-labels. Because each "split" hypercube cluster in a KCube bipartite graph take $\log d + 1$ time to broadcast data,

and all "split" hypercube clusters send or receive data at the same time, all "split" hypercube clusters with different Kautz-part-labels in one KCube bipartite graph take $\log d + 1$ time. Now we can analyse the bipartite protocol that is divided into phases.

During a phase, each vertex which has received the data at some initial/terminal vertex (we use * to represent the next Kauz-part-label on the KCube bipartite graphs, which is the Kautz bipartite partition as described in Section 4.3) of any $K_{d,d}$ sends data to the corresponding terminal/initial vertex of the same $K_{d,d}$ using our bipartite broadcasting protocol. Each KCube bipartite graph takes $\log d + 1$ time units. At phase i, vertices which start a bipartite protocol are of the form $x_i....x_k*$ and none of them can be in the same $K_{d,d}$. After i phases of this protocol, every vertex at a distance of i from the originator has received the data. Because each KCube bipartite graph takes $\log d+1$ time, and because we have $d^{k-1}+d^{k-2}$ number of KCube bipartite graphs, the total running time is the diameter of the KCube bipartite graphs times $\log d + 1$, it is $\Omega(\log((d^{k-1} + d^{k-2}) \times d \times 2^m)) = \Omega(\log((d^{k-1} + d^{k-2}) \times d \times 2d)) = O(k \log d)$. It is optimal running time.

In Figure 4.8, each KCube bipartite graph takes $\log d + 1$ time to send data. There are $d^{k-1} + d^{k-2} = 2^{2-1} + 2^{2-2} = 3$ KCube bipartite graphs. $b(KC(2, 2)) \leq k(\log d + 1) = 3 * 2 = 6$.

In essence, what our bipartite broadcasting algorithm does is allow all informed nodes in "split" hypercube clusters to broadcast data to another "split" hypercube cluster with the same Kautz-part-label in one step simultaneously at one unit of time; therefore, the "split" hypercube clusters are acting as if they were nodes in a Kautz graph.

Chapter 5

Conclusion

In this thesis, we have studied the KCube which has been proposed as a novel architecture for interconnection networks. We found some useful topological properties of the KCube. We also designed an improved routing algorithm and two broadcasting algorithms for this network. In particular, we have proven:

- The KCube possesses the property of bipartiteness, which allows partitioning all vertices into two disjoint sets. These two sets have equal cardinality. Therefore, the KCube is a balanced bipartite graph.
- $KC(1, k)$ and $KC(2, k)$, for arbitrary k, are Hamiltonian.
- The KCube is regular but not vertex-symmetric.

We have designed:

- an improved routing algorithm showing a reduced upper bound on the diameter of the KCube.

- two single-port broadcasting algorithms which are based on Bermond and Perennes's Kautz broadcasting algorithm. These broadcasting algorithms match the lower bound, thus are optimal. The idea is to broadcast data on a number of bipartite graphs. Bipartite graphs are produced by line graphs. The line graph and the original graph with the same degree are isomorphic to each other.

So far, only a few algorithms and topological properties have been found for the KCube. More work still needs to be done to find algorithms and topological properties for solving more problems on the KCube in the future. Some of these problems are listed below:

- designing application algorithms for the KCube (such as sorting).
- finding fault-tolerance properties and other properties for the KCube.
- Hamiltonianicity for $KC(m, k)$ for arbitrary m and k.

Bibliography

[1] S. G. Akl, *Parallel Computations: Models and Methods.* Upper Saddle River, NJ, USA: Prentice Hall, 1997.

[2] S. B. Akers and B. Krishnamurthy, "A Group-Theoretic Model for Symmetric Interconnection Networks," *IEEE Transactions on Computers,* vol. 38, no. 4, pp. 555-566, 1989.

[3] D. P. Agrawal, *Advanced Computer Architecture*. IEEE Computer Society Press, 1986.

[4] J. C. Bermond and S. Perennes, "Efficient Broadcasting Protocols on the de Bruijn and Similar Networks,"in *Proc. 2nd Colloquium on Structural Information and Communication Complexity*, 1995, pp. 233-247.

[5] J. C. Bermond and C. Peyrat, "De Bruijn and Kautz Networks: A Competitor for the Hypercube,"in *Proc. 1st European Workshop on Hypercubes and Distributed Computers*, 1989, pp. 279-293.

[6] D. P. Bertsekas, C. Ozveren, G. D. Stamoulis, P. Tseng, and J. N. Tsitsiklis, "Optimal Communication Algorithms for Hypercubes, " *Journal of Parallel and Distributed Computing*, vol. 11, pp. 263-275, 1991.

[7] N. Biggs, *Algebraic Graph Theory*. Cambridge University Press, 1974.

[8] W. G. Bridges and S. Toueg, "On the Impossibility of Directed Moore Graphs," *Journal of Combinatorial Theory, Series B*, vol. 29, pp. 339-341, 1980.

[9] J. M. Brunat, "Explicit Cayley Covers of Kautz Digraphs," *The Electronic Journal of Combinatorics,* vol. 18, pp. 105, 2011.

[10] C. Chen, D. P. Agrawal, and J. R. Burke, "dBCube: A New Class of Hierarchical Multiprocessor Interconnection Networks with Area Efficient Layout," *IEEE Transactions on Parallel and Distributed System*, vol. 4, no. 12, 1993.

[11] B. C. Cheng, K. S. -M. Li, and S. J. Wang, "De Bruijn Graph-Based Communication Modelling for Fault Tolerance in Smart Grids," *Circuits and Systems (APCCAS) on Asia Pacific Conference*, 2012, pp. 623-626.

[12] W. K. Chiang and R. J. Chen, "Distributed Fault-Tolerant Routing in Kautz Networks,"in *Proc. of the Third Workshop on Future Trends of Distributed Computing Systems*, 1992, pp. 297-303.

[13] J. J. Cook and C. Zilles, "Characterizing and Optimizing the Memory Footprint of de Novo Short Read DNA Sequence Assembly", *Performance Analysis of Systems and Software. IEEE International Symposium on Performance Analysis of Systems and Software*, Urbana, 2009, pp. 143-152.

[14] D. E. Culler, J. P. Singh and A. Gupta *Parallel Computer Architecture A Hardware/Software Approach.* Morgan Kaufmann Publishers, INC, An Imprint of Elsevier, San Francisco, California, 2011.

[15] S. P. Dandamudi and D. L. Eager, "On Hierarchical Hypercube Multicomputer Interconnection Network Design," *Journal of Parallel and Distributed. Computer* vol. 12, no. 3, pp. 283-289, 1991.

[16] D. Z. Du, D. F. Hsu, F. K. Hwang, and X. M. Zhang, "The Hamiltonian Property of Generalized de Bruijn Digraphs," *Journal of Combinatorial Theory, Series B*, vol. 52, pp. 1-8, 1991.

[17] M. Flynn, "Some Computer Organizations and their Effectives," *IEEE Transactions on Computers,* vol. C-21, pp. 948-960, 1972.

[18] P. Fraigniaud and P. Gauron, The Content-Addressable Network D2B. Tech Rept, 2003.

[19] H. Frank, "The Maximum Connectivity of a Graph,"in *Proc. of the National Academy of Sciences of the United States of America,* vol. 48, no. 7, 1962, pp. 1142-1146.

[20] D. Guo, H. Chen, Y. He, H. Jin, C. Chen, H. Chen, Z. Shu, G. Huang, "KCube: A Novel Architecture for Interconnection Networks," *Information Processing Letters*, vol. 110, no. 18-19, pp. 821-825, 2010.

[21] R. Harbane and M. C. Heydemann, "Efficient Reconfiguration Algorithms of de Bruijn and Kautz Networks into Linear Arrays," *Journal of Theoretical Computer Science*, vol. 263, pp. 173-189, 2001.

[22] J. P. Hayes, T. N. Mudge, and Q. F. Stout, "Architecture of a Hypercube Supercomputer,"in *Proc. International Conference on Parallel Processing*, 1986, pp. 653-660.

[23] M. C. Heydemann and D. Sotteau, "A Note on Recursive Properties of the de Bruijn, Kautz and FFT Digraphs," *Information Processing Letters*, vol. 53, pp. 255-259, 1995.

[24] W. D. Hillis, *The Connection Machine.* Cambridge, MA: MIT Press, 1985.

[25] M. F. Kaashoek and D. R. Karger, "Koorde: A Simple Degree-Optimal Hash Table," in *Proc. of the 2nd International Workshop on Peer-to-Peer System(IPTPS)*, 2003.

[26] E. Kranakis and D. Krizanc, "Distributed Computing on Cayley Networks," *4th IEEE Symposium on Parallel and Distributed Processing,* Arlington, pp. 222-229, 1992.

[27] E. Kranakis and D. Krizanc, "Labeled versus Unlabeled Distributed Cayley Networks," *1st Colloquium on Structural Information and Communication Complexity (SICC-1),* Carleton University, Press, 1994.

[28] T. Lakshman and V.K. Wei, "Distributed Computing on Regular Networks with Anonymous Nodes, "*IEEE Transactions on Computers*, vol. 43, no. 2, pp. 211-218, 1994.

[29] C. Lavault, "Interconnection Networks: Graph- and Group- Theoretic Modelling", in *Proc. of 12th International Conference on Control Systems and Computer Science* vol. 2, 1999, pp. 207-214.

[30] D. Li, X. Lu, and J. Su, "Graph-Theoretic Analysis of Kautz Topology and DHT Schemes,"*IFIP International Federation for Information Processing*, 2004, pp. 308-315.

[31] G. Panchapakesan and A. Sengupta, "On a Lightwave Network Topology Using Kautz Digraphs,"*IEEE Transactions on Computers*, vol. 48, no. 10, pp. 1131-1138, 1999.

[32] K. Padmanabhan, "Cube Structure for Multiprocessors,"in *Communications of the ACM,* vol. 33, no. 1, pp. 43-52, 1990.

[33] K. Qiu, S. G. Akl and H. Meijer, "On Some Properties and Algorithms for the Star and Pancake,"*Journal of Parallel and Distributed Computing*, vol. 22, pp. 16-25, 1994.

[34] C. P. Ravikumar, T. Rai, V. Verma, *Kautz Graphs as Attractive Logical Topologies in Multihop Lightwave Networks*. Computer Communications, vol. 20, pp. 1259-1270, 1997.

[35] Y. Saad and M. H. Schultz, "Topological Properties of Hypercubes," *IEEE Transactions on Computers*, vol. 37, no. 7, pp. 867-872, 1988.

[36] M. R. Samatham and D. K. Pradhan, "The de Bruijn Multiprocessor Network: A Versatile Parallel Processing and Sorting Network for VLSI," *IEEE Transactions on Computers*, vol. 38, no. 4, pp. 567-581, 1989.

[37] C. L. Seitz, "The Cosmic Cube," *in Communications of the ACM*, vol. 28, no. 1, pp. 22-33, 1985.

[38] K. N. Sivarajan and R. Ramaswami, "Lightwave Networks Based on de Bruijn Graphs," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 70-79, 1994.

[39] K. W. Tang and B. W. Arden, "Vertex-Transitivity and Routing for Cayley Graphs in GCR Representations," *Proc. of the 1992 ACM Symposium on Applied Computing,* vol. 2, pp. 1180-1187, 1992.

[40] K. W. Tang and B. W. Arden, "Representations of Borel Cayley Graphs," *SIAM Journal on Discrete Math* vol. 6, no. 4, pp. 665-676, 1993.

[41] Wati, "Kautz graph", manuscript, online at http://planetmath.org/sites/default/files/texpdf/38526.pdf, 2013.

[42] D. B. West, *Introduction to Graph Theory.* Upper Saddle River, NJ 07458, USA: Prentice Hall, 1996.

[43] Z. Zhang and A. S. Acampora, "Analysis of Multihop Lightwave Networks," *Proc. IEEE GLOBECOM* pp. 1873-1879, 1990.