2003

Efficient embedding of virtual hypercubes in irregular WDM optical networks

Guru Prasad P. Kithlanagamangala
Louisiana State University and Agricultural and Mechanical College

EFFICIENT EMBEDDING OF VIRTUAL HYPERCUBES IN IRREGULAR WDM OPTICAL NETWORKS

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
In partial fulfillment of the
Requirements for the degree of
Master of Science of Electrical Engineering

In

The Department of Electrical Engineering

By
Guru Prasad P. Kithlanagamangala
Bachelor of Engineering in Telecommunications
Sir M.V.I.T., Bangalore University, 2001
December 2003

Dedicated to my dear parents

Acknowledgements

I take this opportunity to sincerely thank my adviser, Dr. Ahmed El-Amawy for his guidance, ideas and comments throughout the course of this work. He was always available with his time and patiently guided me though the process. I also would like to thank the members of my committee, Dr. Hsiao-Chun Wu and Dr. Jerry L. Trahan. I extend my gratitude to Mr. Sateesh Chandra Shekhar for his help and efforts in implementing software for the algorithms.

I would like to thank the Department of Electrical and Computer Engineering for their financial support through a Teaching Assistantship.

I thank my parents, Mr. K.S. Parameswaraiah and Mrs. N. Lakshmi Devi and my sisters, Sushma Sudhakar and Surma Vallish, for their continued support and encouragement in the course of doing my Masters degree.

Last but not the least, I thank everyone for their support in the successful completion of this work.

Table of Contents

List of Tables

List of Figures

Abstract

This thesis addresses one of the important issues in designing future WDM optical networks. Such networks are expected to employ an all-optical control plane for dissemination of network state information. It has recently been suggested that an efficient control plane will require non-blocking communication infrastructure and routing techniques. However, the irregular nature of most WDM networks does not lend itself to efficient non-blocking communications. It has been recently shown that hypercubes offer some very efficient non-blocking solutions for, all-to-all broadcast operations, which would be very attractive for control plane implementation. Such results can be utilized by embedding virtual structures in the physical network and doing the routing using properties of a virtual architecture. We will emphasize the hypercube due to its proven usefulness.

In this thesis we propose three efficient heuristic methods for embedding a virtual hypercube in an irregular host network such that each node in the host network is either a hypercube node or a neighbor of a hypercube node. The latter will be called a “satellite” or “secondary” node. These schemes follow a step-by-step procedure for the embedding and for finding the physical path implementation of the virtual links while attempting to optimize certain metrics such as the number of wavelengths on each link and the average length of virtual link mappings. We have designed software that takes the adjacency list of an irregular topology as input and provides the adjacency list of a hypercube embedded in the original network. We executed this software on a number of irregular networks with different connectivities and compared the behavior of each of the three algorithms. The algorithms are compared with respect to their performance in trying to optimize several metrics. We also compare our algorithms to an already existing algorithm in the literature.

Chapter 1

Introduction

Fiber optic communications have provided us with high-speed communications with enormous bandwidth potential. Although fibers can support very high data rates, the associated electronic processing hardware will typically not be able to keep up with such speeds. Hence electronic handling of data network nodes basically limits the throughput of the network. Further, electronic processing is required because optical storage and processing technologies are not mature yet. Hence a packet that must be stored or processed at an intermediate node has to be converted to its electronic form and stored in an electronic buffer memory. The header is then extracted, processed and a routing decision is made based on the information provided in the header and the routing protocol. The packet is then queued at the output port, converted back into its optical form and transmitted towards its final destination.

To improve the throughput of the network and to minimize transmission delay, the network architecture must both reduce the number of times a message is processed by the intermediate nodes and must streamline the processing at each node. The irregular nature of most of the existing networks doesn't necessarily allow this. Hence complex routing tables are often used to make routing decisions requiring complicated and time-consuming processing. If the network could be connected in a regular uniform pattern, routing decisions could be significantly simplified thereby reducing the processing times at the intermediate nodes. But because of many real world constraints, a regular uniform pattern in building a network may not be feasible. Also economic reasons necessitate reuse of existing fiber connections in networks, which restricts the physical topology options.

1.1Fundamentals of Optical Communication Systems

Light has an information carrying capacity 10,000 times greater than the highest radio frequencies [3]. Advantages of optical fibers over copper transmission line include the ability to carry signals over long distances, low error rates, immunity to electrical interference, security, and light weight [3, 5].

Fiber optic communication had been experimentally tested in the nineteenth century. However the technology began to advance rapidly and was being used in practical networks in the second half of the twentieth century. After the viability of transmitting light over fiber had been established, the next step in the development of fiber optics was to find a light source that would be sufficiently powerful and narrow. Light emitting diodes (LED) and laser diodes (LD) proved capable of meeting these requirements. Researchers in the mid 1960s proposed that optical fiber might be a suitable transmission medium. There was an obstacle, however, and that was the loss of signal strength, or attenuation, seen in the glass with which they were working. Finally, in 1970, Corning produced the first communication-grade fibers [3]. With attenuation less than 20 decibels per kilometer (dB/km), this purified glass fiber exceeded the threshold for making fiber optics a viable technology [5].

Further developments in fiber optics are closely tied to the use of the specific regions on the spectrum where optical attenuation is low. These regions, called windows, lie between areas of high absorption. The earliest systems were developed to operate around 850 nm, the first window in silica-based optical fiber. A second window (S band), at 1310 nm, soon proved to be superior because of its lower attenuation, followed by a third window (C band) at 1550 nm with an even lower optical loss. Today, a fourth window (L band) near 1625 nm is under

development and early deployment. These four windows are shown relative to the electromagnetic spectrum in Figure 1.1 [5] and Figure 1.2 [5].

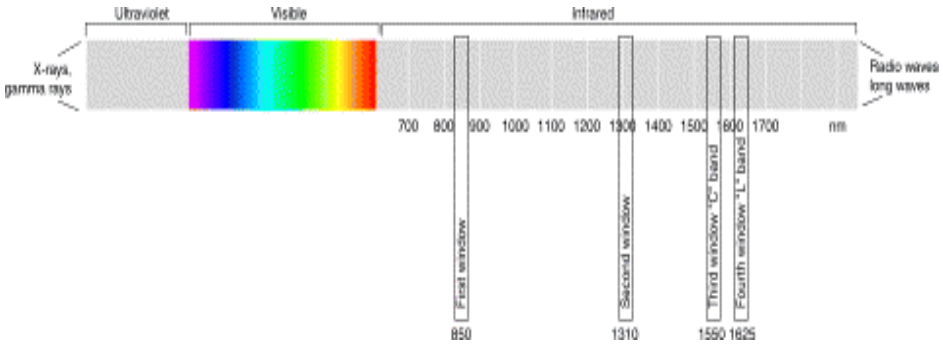


Figure 1.1: Wavelength regions

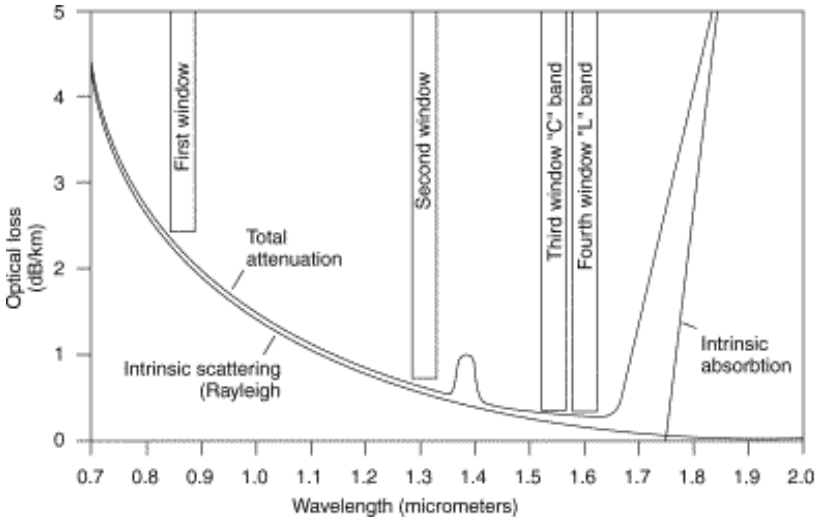


Figure 1.2: Total attenuation curve

A basic optical communication system consists of an optical transmitter, an optical receiver and optical fiber as the communication medium. There are several other components that go into the system to make it practical, such as optical add/drop multiplexers, optical amplifiers, switches and wavelength converters, which have made Dense Wavelength Division Multiplexing (DWDM) feasible.

1.1.1 Optical Fiber

The main purpose of an optical fiber is to guide light waves with minimum attenuation (loss of signal). Optical fibers are composed of fine threads of glass in layers, called the core and cladding that can transmit light at about two-thirds the speed of light in vacuum. Though admittedly an oversimplification, the transmission of light in optical fiber is commonly explained using the principle of total internal reflection. With this phenomenon, 100 percent of light that strikes a surface is reflected. Light is either reflected or refracted depending on the angle of incidence (the angle at which light strikes the interface between an optically denser and optically thinner material). Total internal reflection happens when the following conditions are met.

- The beams pass from a denser to a less dense material. The difference between the optical density of a given material and a vacuum is the material's refractive index.
- The incident angle is less than the critical angle. The critical angle is the maximum angle of incidence at which light stops being refracted and is instead totally reflected.

The principle of total internal reflection within a fiber is illustrated in Figure 1.3 [5]. The core has a higher refractive index than the cladding, allowing the beam that strikes that surface at less than the critical angle to be reflected. The second beam does not meet the critical angle requirement and it refracted.

1.1.2 Light Sources and Detectors

Light sources and light detectors are active devices at opposite ends of an optical transmission system. Light sources, or light emitters, are transmit-side devices that convert electrical signals to light. The process of this conversion, or modulation, can be accomplished

by externally modulating a continuous wave of light or by using a device that can generate modulated light directly. Light detectors perform the opposite function of light emitters. They are receive-side opto-electronic devices that convert light pulses into electrical signals.

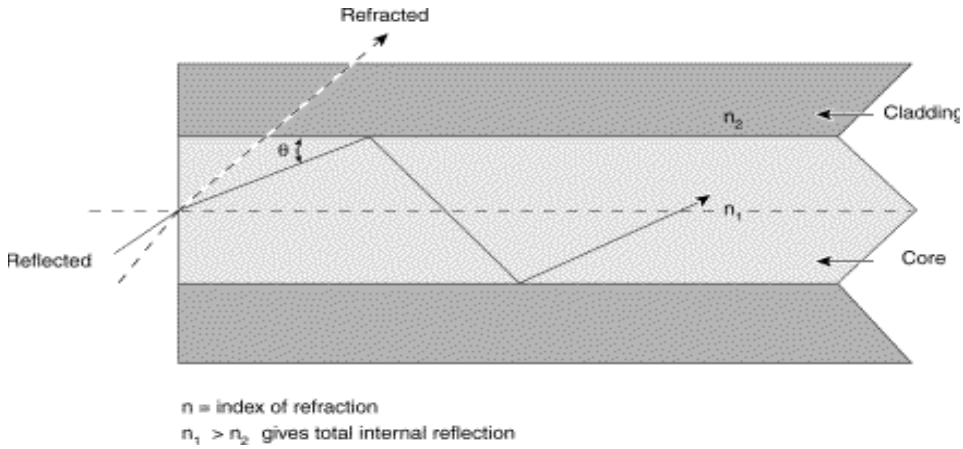


Figure 1.3: Principle of total internal reflection

1.1.3 Optical Amplifiers

Optical amplifiers are important components of an optical communication system that have tremendously increased the distance of propagation of an optical signal with integrity without the need for any Optical-Electronic-Optical (O-E-O) conversion for the purpose of regeneration of the signal. The optical amplifiers (OAs) have made it possible to amplify all the wavelengths at once and without O-E-O conversion. Besides being used on optical links, optical amplifiers also can be used to boost signal power after multiplexing or before demultiplexing, both of which can introduce loss into the system.

The Erbium-Doped Fiber Amplifier (EDFA) was one of the major breakthroughs in the field of optical communications and particularly was a key enabling technology for DWDM optical networks. Erbium is a rare-earth element that, when excited, emits light around 1.54

micrometers, the low-loss wavelength for optical fibers used in DWDM [2]. Figure 1.4 [5] shows a simplified diagram of an EDFA. A weak signal enters the erbium-doped fiber into which light at 980 nm or 1480 nm is injected using a pump laser. This injected light stimulates the erbium atoms to release their stored energy as additional 1550 nm light. As this process continues down the fiber, the signal grows stronger. The spontaneous emissions in the EDFA also add noise to the signal; this determines the noise figure of an EDFA.

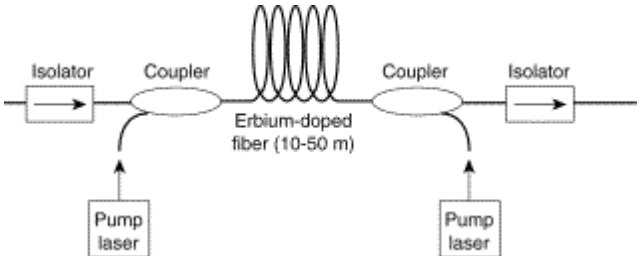


Figure 1.4: Erbium Doped Fiber Amplifier Design

1.1.4 Multiplexers and Demultiplexers

These components are essentially a part of DWDM systems, wherein signals are sent from several sources over a single fiber and are destined to several destinations. The component that combines incoming signals from different sources on a single fiber or takes optical wavelengths from multiple fibers and converges them into one beam is called a multiplexer. At the receiving end, the system must be able to separate out the components of the light so that they can be discreetly detected. Demultiplexers perform this function by separating the received beam into its wavelength components and coupling them to individual fibers. Demultiplexing must be done before the light is detected because photo detectors are inherently broadband devices that cannot selectively detect a single wavelength.

In a unidirectional system (Figure 1.5), there is a multiplexer at the sending end and a demultiplexer at the receiving end. Two systems would be required at each end for bi-

directional communication, and two separate fibers would be needed. In a bi-directional system (Figure 1.6), there is a multiplexer/demultiplexer at each end and communication is over a single fiber with different wavelengths.

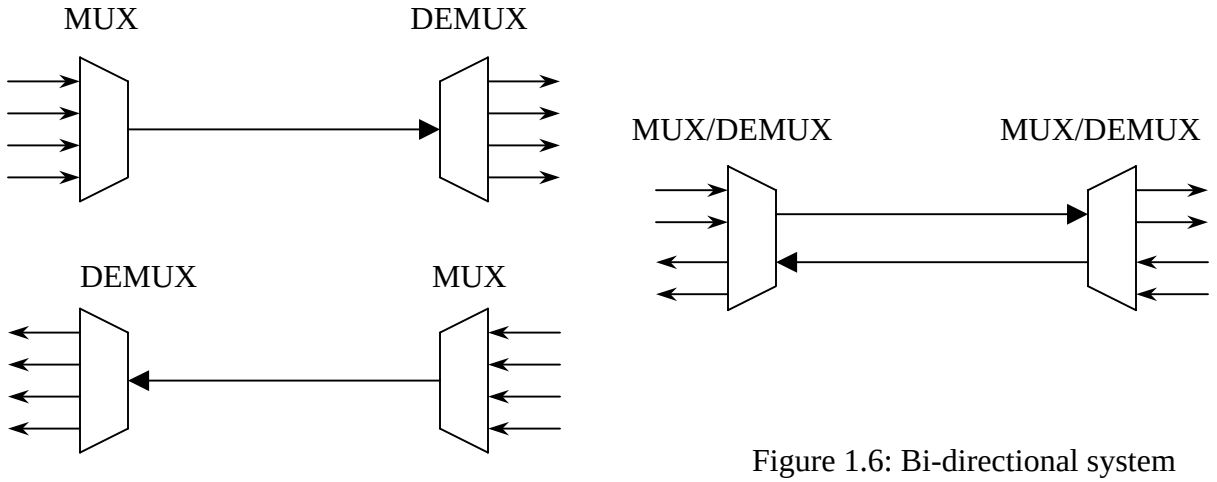


Figure 1.6: Bi-directional system

Figure 1.5: Unidirectional system

1.1.5 Optical Add/Drop Multiplexers

Between multiplexing and demultiplexing points in a DWDM system, as shown in the above figures, there is an area in which multiple wavelengths exist. It is often desirable to be able to remove or insert one or more wavelengths at some point along this span. An optical add/drop multiplexer (OADM) performs this function. Rather than combining or separating all wavelengths, the OADM can remove some while passing on others. OADMs are a key part of moving toward the goal of all-optical networks [6]. OADMs are similar in many respects to SONET ADM, except that only optical wavelengths are added and dropped, and no conversion of the signal from optical to electrical takes place. Figure 1.7 is a schematic representation of the add-drop process. This example includes both pre- and post-amplification; these components that may or may not be present in the OADM, depending upon its design.

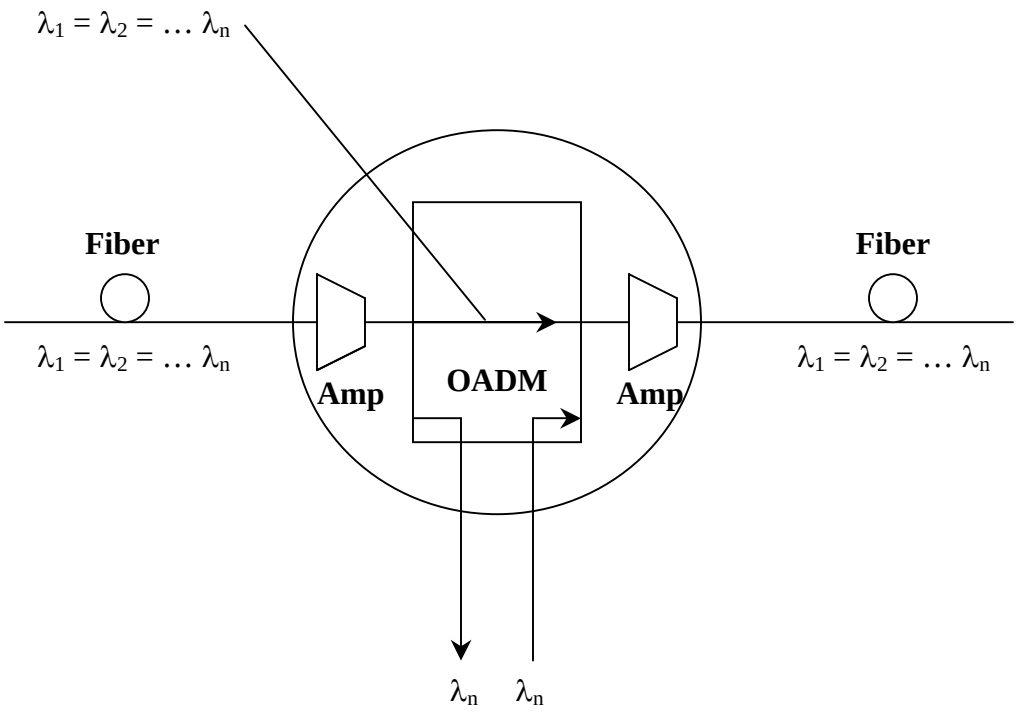


Figure 1.7: Selectively removing and adding wavelengths

1.1.6 Wavelength Converters

Wavelength converters are now becoming important components of all optical networks (AONs). As the name suggests, a wavelength converter does a conversion of an incoming wavelength λ_1 to an outgoing wavelength λ_2 without any optical-electronic-optical conversion. There are many different types of wavelength converters that have varying capabilities. A wavelength converter can have full-scale capability wherein any incoming wavelength can be converted into any other wavelength used in the network, or it can have limited conversion capability where a wavelength can be converted into a subset of the total

number of wavelengths being used. As wavelength converters are expensive, their placement is a critical issue in many networks [23].

1.2 Wavelength Division Multiplexing (WDM) and Dense Wavelength Division Multiplexing (DWDM)

Wavelength division multiplexing (WDM) increases the carrying capacity of the physical medium (fiber). In a WDM system, incoming optical signals are assigned to specific frequencies of light (wavelengths, or lambdas) within a certain frequency band and are transmitted on the same fiber medium. At the receiving end these wavelengths are demultiplexed using detectors tuned at the specific frequencies or wavelengths. The term *wavelength* is used instead of the term frequency to avoid confusion with other uses of frequency. Wavelength is often used interchangeably with *lambda* and *channel*. A simple schematic of a WDM system is shown in Figure 1.8 [5].

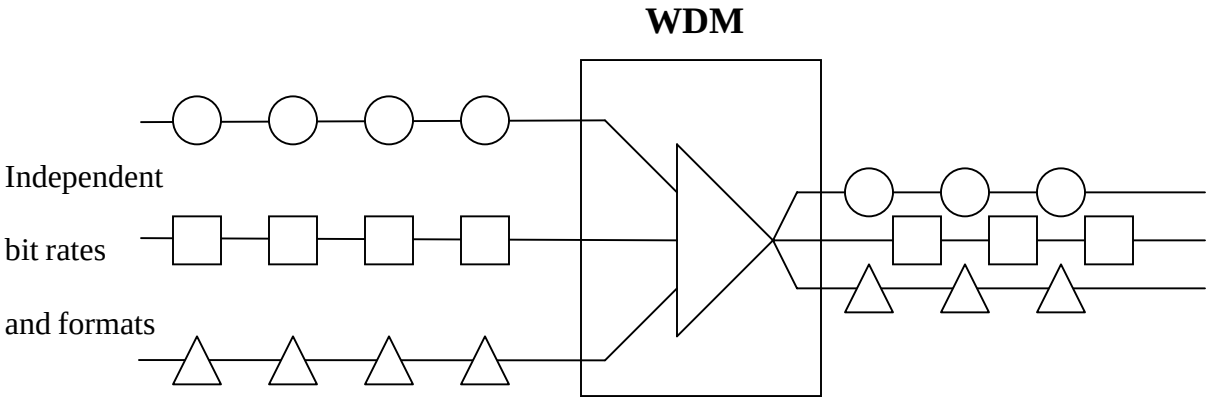


Figure 1.8: Schematic of a WDM system

In a WDM system, each input signal is carried independently of the others. This means that each channel has its own dedicated bandwidth; all signals are sent at the same time, rather than being broken up and carried in time slots as in the case of a Time Division Multiplexed (TDM) signal.

The difference between WDM and dense wavelength division multiplexing (DWDM) is fundamentally one of only degree. DWDM spaces the wavelengths more closely than does WDM and therefore has a greater overall capacity. The limits of this spacing are not precisely known, and have probably not been reached, though systems are available with a capacity of 128 lambdas on one fiber.

From both technical and economic perspectives, the ability to provide potentially unlimited transmission capacity is the most obvious advantage of DWDM technology. The current investment in fiber infrastructure cannot only be preserved, but also be optimized by a factor of at least 32, where 32 is the minimum number of wavelengths which can be sent over a single fiber using DWDM. As demands change, more capacity can be added, either by simple equipment upgrades or by increasing the number of lambdas on the fiber, without expensive upgrades. Capacity can be obtained for the cost of the equipment, and existing fiber plant investment is retained.

Bandwidth aside, DWDM's most compelling technical advantages can be summarized as follows:

- Transparency—Because DWDM is a physical layer architecture, it can transparently support both TDM and data formats such as ATM, Gigabit Ethernet, ESCON, and Fiber Channel with open interfaces over a common physical layer.
- Scalability—DWDM can leverage the abundance of dark fiber in many metropolitan area and enterprise networks to quickly meet demand for capacity on point-to-point links and on spans of existing SONET/SDH rings.

- Dynamic provisioning—Fast, simple and dynamic provisioning of network connections give providers the ability to provide high-bandwidth services in minutes rather than hours.

The optical components discussed earlier like erbium-doped fiber optical amplifiers, optical add/drop multiplexers, wavelength converters and low attenuation optical fibers have provided the enabling technologies for the practical implementation of DWDM networks.

1.3 Routing in WDM Optical Networks

Routing and wavelength assignment (RWA) is another important aspect in implementation of optical WDM networks. Routing in optical networks is similar to circuit switched electronic networks and is performed by generating lightpaths. A lightpath is a path taken by an optical signal between a source and a destination. The wavelength occupied by a lightpath could either use a single wavelength, or it could use multiple wavelengths with wavelength converters at intermediate nodes. The RWA problem deals with routing and assigning wavelengths at every hop in the path between a source-destination pair [25]. RWA schemes can be classified into two categories:

- Offline or static schemes and
- Online or dynamic schemes.

In a static RWA scheme, all the routes and wavelengths for all lightpaths between different source-destination pairs are fixed a *priori*. Whenever a request for a connection arrives, the RWA scheme assigns the pre-allocated lightpath for that request. Therefore the routing

procedure does not change with time, and it is not complex to implement. Dimension order routing is an example of static routing for regular topologies like mesh, hypercubes, etc. [3]. In this technique, while routing a connection request, a dimension is completely exhausted before going to the next dimension. A dynamic RWA scheme uses the current state of the network to determine the route for a given lightpath request. Therefore, the route chosen to establish a connection reflects the current utilization of the links and the status of the network. A connection is blocked if there is no available route at that instant. Dynamic RWA schemes can be further classified into two types based on whether they require global or local information to make a routing decision. Adaptive routing approaches are examples for dynamic RWA schemes.

One of the important parameters for an RWA algorithm is to minimize the blocking probability. Blocking probability is an important measure of how efficient the routing algorithm is for a network and is defined as the probability of a call being blocked due to the unavailability of sufficient resources. Studies have shown that dynamic RWA schemes tend to provide a higher blocking probability when compared to static schemes without wavelength converters [16].

A control plane is essentially a shadow network of the original network with the same or a different topology. It takes care of the exchange of control messages between the nodes of the network for smooth data flow within the data network. Routing in the control plane can be in-band or out-of-band. In-band control essentially means reserving a set of wavelengths for the sole purpose of routing control information. In practical networks, normally the control network uses 5% of the total number of wavelengths. Out-of-band control refers to a

completely different network that has the same or different topology as that of the original data network and which is used for transmitting control messages. Even though the network topologies of the data and the control networks can vary, they both typically have the same number of nodes.

Routing in the control plane is needed for the purpose of global exchange of control information between all the nodes in the network. This information is commonly referred to as link status advertisement (LSA). Every node has to communicate its link status periodically to every other node in the network. Every node also has to broadcast its LSA whenever there is a change in the status of its links, such as a link going down or a link coming up. This all-to-all broadcast consumes a sizeable amount of the network bandwidth. In a recent study it has been proposed that for control plane implementation such all-to-all broadcast should be non-blocking since smooth functioning of the data network depends on the fast and reliable exchange of LSAs [17, 32]. This exchange of LSAs can be performed more efficiently with regular networks like ring, mesh, torus, hypercube, etc. These structures by virtue of their geometry offer several advantages that can be harnessed to route the link status packets with no or minimal processing across the network. On the other hand, with irregular networks, a huge amount of bandwidth and processing time are required for periodic transmission of LSAs between all the nodes of the network. In case of dynamic RWA schemes, every node has to make routing decisions depending on the current network status. This increases the overhead in terms of the processing delay. If static RWA schemes are used, large amount of wavelengths have to be reserved to achieve non-blocking routing which means that we are adding a considerable amount of overhead in terms of network bandwidth utilization hence reducing the

overall throughput of the network. A simple solution to the problem of routing link status information efficiently within the network is to utilize the properties of regular networks to work for irregular networks in order to simplify routing. This can be achieved by embedding the regular structure within the irregular network. The regular structure is treated as virtual network embedded within the original physical topology. Once such an embedding is developed, routing can be performed within the virtual regular structure utilizing all its properties to simplify the exchange of link status information within the network.

This thesis deals with the embedding of a WDM hypercube in WDM optical network with an arbitrary topology for the purpose of simplifying the routing. The motivation behind the work and some previous work is discussed in Chapter 2. Three heuristic algorithms for the problem of efficiently embedding hypercubes in irregular WDM networks are introduced in Chapter 3. Chapter 4 reports the results and observations on the performance and behavior of the proposed algorithms. The final chapter comprises a conclusion and an insight into possible future enhancements.

Chapter 2

Background and Literature Review

The potential for achieving very high data rates has increased multifold by the advent of wavelength division multiplexing (WDM) where a single fiber can carry a number of wavelengths. This technology can be used to create a virtual topology within an irregular physical network. WDM provides multiple independent logical channels on each physical link. A virtual topology can be embedded into a physical topology by establishing logical (or virtual) connections. Two non-adjacent nodes in the "host network" can be connected by a lightpath to form a logically adjacent pair of nodes in the "guest" network. Hence in a virtual topology, we could define virtual paths, which are essentially a wavelength or a set of wavelengths reserved from a source node to a destination node. Reserving wavelengths could be done online or offline. Offline wavelength assignments could be used where a wavelength is fixed for a particular lightpath and would essentially require no processing at intermediate nodes (they would be bypassed using add/drop multiplexers). This provides a direct optical connection between non-adjacent nodes. Light signals on these channels or lightpaths would travel without any O-E-O conversion switching or processing delays. So, instead of having a direct optical connection only to those nodes that are physically adjacent, each different WDM channel can provide a direct optical connection to a different node. Hence the interconnection of WDM channels provides a virtual topology that is independent of the actual physical topology of the network.

Since the virtual topology can be made independent of the physical connections, the optical channels can be connected to form any one of the many different virtual (regular) topologies, such as trees, rings, 2-dimensional meshes, hypercubes, etc. This process of establishing a virtual topology is called e*mbedding*. Such embedding can be utilized very

efficiently to perform complex communication patterns with little or no conflicts [9]. The choice of virtual topology affects the complexity of routing, message path length and the amount of hardware required to implement the topology. An effective embedding should have a small dilation and short average path length. Other factors to consider include the number of wavelengths dedicated to the embedding, average length of a virtual link, … etc.

2.1 Hypercube

A *d*-dimensional hypercube consists of 2^d nodes and a d + 1 dimensional hypercube can be constructed by connecting two *d*-dimensional hypercubes. Each node in a hypercube is assigned a *d* bit binary label. Some examples of different dimensional hypercubes are shown in Figure 2.1.

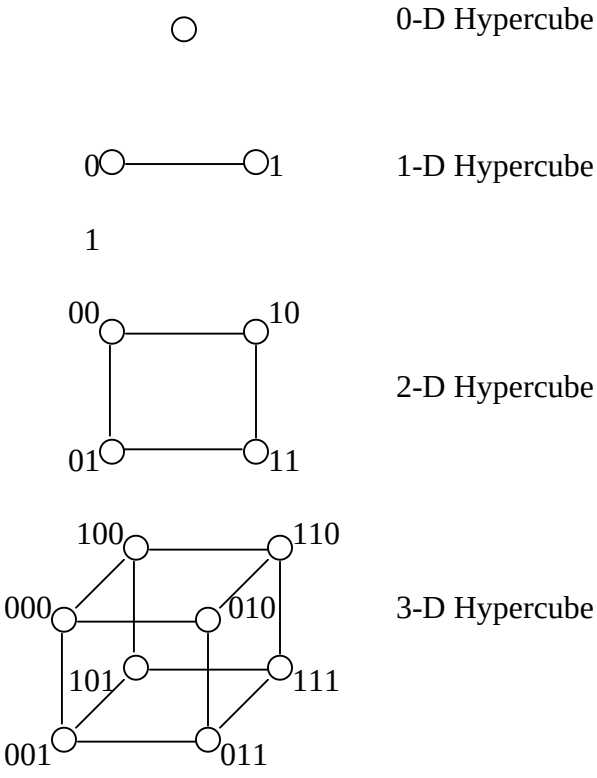


Figure 2.1: Hypercubes

2.1.1 Properties of Hypercubes

Some of the properties of a hypercube [19] are as follows.

- Two nodes are connected by a direct link if and only if the binary representations of their labels differ at exactly one bit position.
- In a *d*-dimensional hypercube, each node is directly connected to *d* other nodes.
- A *d*-dimensional hypercube can be partitioned into two d – 1 - dimensional hypercubes.
- Node labels in a *d*-dimensional hypercube contain *d* bits.
- Diameter = *n*.
- Average path length = *n* log (*n*) / [2 (*n*-1)] links.
- Multiple minimum length paths and good number of longer alternate paths.

2.2 Embedding

Given two graphs, *G* (*V*, *E*) and *G`*(*V`*, *E`*), embedding graph *G* into graph *G`*:

- Maps each vertex in *V* onto a vertex (or a set of vertices) in *V`* and
- Maps each edges in *E* onto an edge (or a set of edges) in *E`*.

In traditional networks, the three important embedding parameters are:

- **Congestion**: The maximal number of edges in *E* mapped onto a single edge in *E`*.
- **Dilation**: The maximal number of edges in *E`* that any edge in *E* is mapped onto.
- **Expansion**: The ratio of the number of vertices (processors) in *V`* to that in *V*.

In optical network embeddings, other important parameters must be considered including:

- Number of wavelengths per link
- Average length of virtual links
- Mean physical path length

2.3 Motivation

The most important motivation behind this work is to find simpler, well-defined ways for routing on the control plane of an irregular network. We try to build upon recent results [17] in non-blocking routing algorithms to enable the implementation of extremely efficient global information exchange. This is very desirable. To utilize the results in [17], we address the issue of embedding a regular structure (hypercube) into a WDM network with arbitrary topology with special attention to optical network characteristics.

Apart from the above stated properties, hypercubes are well suited for embedding into irregular networks because routing can be extremely simple. Routing within a hypercube is essentially done by bit correction. The Hamming distance and message routing between nodes *s* and *t* can be described as follows. Let $r = s \oplus t$ be the bitwise exclusive-OR operation on *s* and *t*. Let $|r|$ be the number of ones in *r*. The number of communication links between the two processors *s* and *t* is equal to $|r|$ or the hamming distance between their labels. Message routing from *s* to *t* can be done by passing a message along the dimensions that correspond to 1 bits in $s \oplus t$, or in other words, by just correcting the bits in each dimension in which the labels of *s* and *t* differ.

Since there exist efficient methods [16, 17] that utilize offline fixed wavelength assignments with no wavelength converters; no or little processing is required at the intermediate nodes. Wavelengths are pre-assigned between two nodes, which may be connected by either a single physical link (which makes such nodes neighbors in both physical and virtual topologies), or by a virtual link (or lightpath) having intermediate nodes. In the latter case, wavelengths are bypassed by the particular intermediate node with the help of add/drop multiplexers.

Embedding of a virtual hypercube in an arbitrary irregular physical network involves assigning virtual addresses to each physical node and defining a path through the physical network to connect nodes that are adjacent in the virtual topology. Also since we consider the number of physical nodes as always greater than the number of nodes in the virtual hypercube, we will have certain nodes called *"satellite or secondary nodes"* connected to the primary nodes. We impose the constraint that any given satellite node is at a maximum distance of 1 from a primary node in the physical topology. The number of satellite nodes connected to a primary node is kept to a minimum and the load across all the primary nodes in terms of the number of satellite nodes connected to the primary node is balanced as much as possible. Once a satellite node is connected to a primary node, it becomes the responsibility of the primary node to transmit all the routing, control and link status information of the network to and from its satellite node(s).

2.4 Previous Work

Although embedding hypercubes in irregular WDM networks has been the subject of a few studies [9, 10, 13], considerable research has been done in the past on the reverse problem of embedding arbitrary networks in hypercubes [18, 26, 27]. One of the first embedding attempts was made using the ShuffleNet system [28], which uses a perfect shuffle virtual topology embedded in a broadcast physical network. The broadcast network can be in several different forms including a D-bus or a passive star, as long as each transmission is visible to all nodes in the network. In ShuffleNet, a node has only a small number of fixed wavelength transmitters and receivers (typically only two each). Each transmitter in the network operates on a different wavelength. There is one receiver for each wavelength connecting the transmitters and receivers

together in a perfect shuffle topology. Because all transmissions are broadcast on a common channel, ShuffleNet requires the WDM system to support at least 2*n* wavelengths.

Chlamtac *et al.* have defined the Lightnet architecture [29]; a point to point WDM network with an embedded virtual hypercube. They present an algorithm for embedding a hypercube in an arbitrary physical network with 2^n nodes (*n* being the dimension of the hypercube) and prove a lower bound on the number of wavelengths required in the embedding [13]. Their algorithm embeds a string or a linear topology in the physical topology along a Hamiltonian path and a hypercube is then embedded in the string. If the physical network does not contain a Hamiltonian path, an Eulerian path or spanning tree traversal is used. For a network with *n* nodes, the embedding of a hypercube in a string is proved to require no more than 2/3 *n* wavelengths. Because all networks contain a spanning tree and a traversal of the tree describes the desired topology, no irregular topology will require more than 2/3 *n* wavelengths to embed a hypercube. Assuming unit distance between physically adjacent nodes in the string, the average path length for all node pairs is $n^2 - 1/\ 3(n\text{-}1)$ [20].

A three-step approach was followed in another heuristic embedding algorithm [9] where embedding a hypercube in the irregular graph is done such that the number of virtual edges that have a dilation of one is maximized. The first step consists of mapping virtual addresses to the physical nodes, which is done by finding smaller cycles of lower dimensions of the hypercube used in the embedding. This, according to the heuristic, makes the dilation of virtual edges more likely to be one as these smaller cycles are directly embedded onto the hypercube cycles. The second step is to find a path through the physical network for each pair of virtually adjacent nodes minimizing the maximum number of virtual connections on any physical link. The final step is to assign a specific wavelength in the WDM network minimizing the number of

wavelengths required. The major drawback of this heuristic is that it only considers networks which have number of nodes equal to a power of 2; which is seldom the case for real world irregular networks.

Another approach, described in [10], uses unidirectional incomplete hypercubes, which considers the physical networks to be of any size and claims that the performance of this algorithm is comparable to both unidirectional and bi-directional hypercubes. But in the case of current optical networks, considering unidirectional links means a huge amount of fiber not being used.

In this thesis we introduce three new heuristic schemes for efficient embedding of hypercubes in any irregular network with emphasis on optical network properties and performance. The first heuristic tries to directly embed the hypercube onto the physical topology by beginning with the node that has the highest node degree and building the embedding around that node recursively. The second heuristic finds an embedding by finding a primary cycle under certain constraints and placing that cycle on the known Hamiltonian cycle of the virtual hypercube. The third heuristic, which is a variant of the second method, also finds a primary cycle in the physical topology while placing this cycle on the Hamiltonian cycle of the hypercube, but uses different criteria in finding the primary cycle

The complete embedding procedure in all the three methods consists of two steps. The first step consists of finding the actual mapping of the virtual hypercube nodes to the nodes of the irregular physical topology. The second step performs physical realization of the virtual links in the hypercube. Specific techniques are used to minimize of the number of wavelengths, maximize the number of virtual links with dilation 1 and minimize the average virtual path

length, as much as possible. The actual algorithms and the various steps involved are discussed in detail in the next chapter.

Chapter 3

Embedding Algorithms

Embedding of a virtual hypercube in an arbitrary irregular physical network involves assigning virtual addresses to physical nodes and defining a path through the physical network to connect the nodes that are adjacent in the virtual topology. We will always consider the number of physical nodes to be larger than the number of nodes in the virtual hypercube. Thus we will have certain nodes called “satellite” or “secondary” nodes. These nodes will be connected to one of the primary (hypercube) nodes with the constraint that any given satellite node is at a maximum distance of 1 from at least one primary node in the physical topology. The algorithms we present will try to achieve some objectives. One of these objectives is to balance the number of satellite nodes assigned to primary nodes. Once a satellite node is connected to a primary node, it becomes the responsibility of the primary node to communicate all the routing, control and link status information of the network to and from that satellite node.

Before getting into the detailed description of our heuristic algorithms, let us define some important terms, which we will be using throughout the remainder of this thesis.

- Primary Node: Node in the irregular network for which there is a one-to-one correspondence to a node of the hypercube.
- Satellite Node: Node that has no direct one-to-one correspondence to a hypercube node. We require that each satellite node be a direct neighbor of a primary node (physically).
- Adjacency List: List representation of a network. It gives the adjacency information for every node in the network (it defines the neighbors of every node).

- Hyperlist: It is the adjacency list of a hypercube.

3.1 Problem Description

We consider a physical network, consisting of nodes connected in an arbitrary topology. Each link is a pair of unidirectional optical fibers oriented in opposite directions, supporting logical channels through wavelength division multiplexing. This arbitrary physical network can be modeled as a directed graph, G_P (P, E) where P is the set of physical nodes and E is the set of directed edges connecting the nodes. To model the multiple WDM channels, there are "w" parallel edges between each pair of adjacent nodes. There is no restriction on the number of nodes present in P. The guest graph to be embedded is the virtual graph, G_V (V, C), where V is the set of virtual nodes and C is the set of virtual edges. The nodes in V are labeled from 0 to n-1. The length of an edge in C is the sum of the lengths of the physical edges in E that implement the virtual connection. We require that $|P| > |V|$, where $|X|$ refers to the cardinality of set X.

Given a physical graph, G_p, our primary goal is to: 1- find a one to one mapping of virtual addresses V onto the physical nodes P (that is, assign a physical node to each virtual node), and map the remaining nodes in P as satellite nodes to the mapped physical nodes; and 2- map each virtual link in C onto physical edge(s) in E such that the number of the wavelengths used on any given edge in E is minimized (as much as possible).

We define a function f: $V \rightarrow P$ that maps virtual addresses to the physical nodes. The goal is to find a combination of virtual address mapping and realization of the virtual connections that minimizes:

$$\sum_{adjacent\ (v_i,\ vj)} \text{dist}\ (f(v_i), f(v_j)),$$

where *adjacent* (v_i, v_j) represents the set of virtual addresses that are adjacent in the hypercube. The function dist (p_a, p_b) defines the length of the WDM channels used to provide the connections between virtually adjacent nodes v_i and v_j. The length of the physical path between two nodes is defined as:

$$dist\ (p_a, p_b) = d\ (p_a, p_{i1}) + d\ (p_{i1}, p_{i2}) + \ .\ .\ .\ + d\ (p_{ij-1}, p_b) + d\ (p_i, p_b)$$

where $d\ (p_{ik}, p_{ik+1})$ represents the length of physical link (P_{iu}, P_{iv}) on that path. To keep the number of wavelengths required to a minimum, the function must choose shortest disjoint paths as much as possible. If such shortest disjoint paths are not available, then the routing function must try to find disjoint non-shortest paths that are as short as possible. Using this method for finding virtual path implementations we attempt to maximize the usage of shortest paths while minimizing the number of wavelengths used.

3.2 Description of Algorithms

A few algorithms currently exist for embedding a hypercube into an arbitrary graph [9, 10, 13]. However with these algorithms the number of wavelengths and the average path length required are far from optimal. Because the problem of optimally embedding a hypercube into an irregular graph is NP-hard [9], one cannot expect to find an algorithm that produces optimal solutions in all cases. The three heuristic algorithms proposed in this thesis consist of two phases each. In all cases, we choose the largest size hypercube such that $|V| < |P|$ where $|X|$ refers to the cardinality of set X. The first phase finds the actual mapping between virtual nodes and physical nodes. Once such a mapping is developed, the second phase tries to find physical path implementations of the virtual links. This second phase is the same for all three algorithms. Hence we will discuss first the three different algorithms for

mapping the virtual nodes and then describe the second phase that maps the virtual links. As stated earlier, this phase is common to all three algorithms.

3.2.1 Virtual to Physical Address Mapping

The goal of this first phase of the algorithm is to find a mapping of virtual addresses to the physical nodes, f: $V \rightarrow P$, such that the number of edges that map with dilation 1 is maximized. This will provide an embedding that maximizes the number of nodes that are both physically and virtually adjacent. In other words, such a mapping maximizes the number of virtual connections that span only one physical link and will tend to have a relatively low average virtual path length. We have tried to achieve this objective using three different methods namely:

- Direct Method Algorithm
- Cycle Method 1 Algorithm
- Cycle Method 2 Algorithm

These algorithms are discussed in detail in the following sections.

3.2.1.1 Direct Method Algorithm

The Direct Method algorithm is based on the principle of building the embedding around the node with the highest node degree. Here, the node with the highest node degree in the host (irregular) topology is assigned to one of the hypercube nodes. If more than one node has the same “highest” node degree, then the one with a smaller label is selected. Depending on the node degrees of its neighbors, the neighbors are mapped to neighbors of the currently placed hypercube node. We place the neighbors whose node degree is equal to (if any) or is one less than the degree of the node with the highest node degree. If such a neighbor is found, then its neighbors with the same node degree are placed. If such a

neighbor is not found, we process some other node which has a node degree equal to the "current" highest node degree or a node with a node degree one less than the highest node degree. This process is repeated until all the hypercube nodes in *V* are mapped to physical nodes in *P*. The nodes that are not mapped, i.e., {*P*}– {*V*}, become the satellite nodes and have to be placed as satellites to the primary nodes while satisfying the following two constraints.

- Each satellite node is at a maximum physical distance of 1 from a primary node. (We assume that this is always possible)
- The association of satellite nodes with the primary nodes is balanced as evenly as possible.

If an embedding satisfying the above conditions is not found, we find another embedding by changing the order of the neighbors in the adjacency list.

Some of the notations, data structures and procedures used in our algorithm are

N_P:	Total number of nodes in the physical graph.
N_V:	Number of nodes in the virtual hypercube = 2^n, where n is the dimensionality of the hypercube.
Sort[ ]:	A function which sorts the adjacency list of the physical graph according to different fields.
currentNode:	Node whose neighbors are currently being processed.
neighborMaxDegree:	The maximum node degree of the neighbors of the current node.
oudeg(i):	Out degree of node (*i*).
actualDegree:	Out degree of the node being processed.

underProcessingNodes: List of nodes that are under processing. Nodes to be placed are picked up from this list.

probableSatelliteList: Nodes that might be satellite nodes. If a node from the underProcessingNodes list cannot be placed, it is placed in this list. If in the future, a node, which is present in the list, finds a mapping, that node is deleted from this list.

pIndex: Pointer in the underProcessingNodes list that keeps track of the currently processed node.

sIndex: Pointer in the probableSatelliteList that keeps track of the node currently being updated as a probable satellite.

cIndex: Pointer in the underProcessingNodes list that keeps track of the nodes that would be considered after initial placement.

node[i]. adj[j]: Refers to the j^{th} neighbor of node *i*.

encapsulateNodeDetails: Function to attach the degree and adjacency of a node.

Steps of the algorithm can be stated as follows:

Step 1: Examine the host network, read it into a structure and find the total number of nodes (N_p).

Step 2: Determine the dimensionality of the hypercube required, *n*, which satisfies the following: Find the largest n such that $2^n < |P|$. Once n is found we have the set *V* of virtual nodes such that $2^n = |V|$. Set *V* consists of the set of labels from 0 to $2^n - 1$, i.e., $V = \{0, 1, 2, \ldots 2n - 1\}$.

Step 3: For every node in the host graph, sort the list of neighbors according to their node degree.

Step 4: Sort the list of nodes in the host network according their node degrees.

Step 5: Choose the appropriate hyperlist based on *n* found in Step 2.

Step 6: Set currentNode = node with the highest node degree (top of the sorted list of host network structure). In case of a tie, pick first one encountered.

Step 7: *count* = 0; While (*count* < number of nodes in the network) {

Step 7(i): neighborMaxDegree = maximum degree among neighbors of current node

/*Finding the highest node degree among neighbors of the current node*/

Step 7(ii): for (*j*=0; *j*<currentNode.outdeg; *j*++){

actualDegree = actualDegree (*node [i]. adj [j]*);

if (actualDegree >= currentNodeDegree)

/*Condition checking if the node degree is greater than or equal to the current node degree, hence nodes with lower node degree are not mapped at this point*/

{

if (node not mapped)

{

Map it to an available hyperlist node as a neighbor of the latest placed node

UnderProcessingNodes[*pIndex*]=*node[i].adj[j]*;

/*Update under processing nodes list*/

pIndex++;

}

```
                    Update probable list
                    }
                    else
                    {
                        /*Node is absent in the hyperlist structure*/
                        probableSatelliteNodes[sIndex]=node[i].adj[j];
                        /*Update satellite list*/
                        sIndex++;
                    }
                }
                if (pIndex > 0)
                {
                    /*Underprocessing array is non-empty*/
                    currentNode =
                    encapsulateNodeDetails(underProcessingNodes[cIndex]);
                    cIndex ++;
                }
                else
                {
                    Use an element from the satellite list
                }
                Update the variable count from the hyperlist
}
```

Step 8: If there are unprocessed nodes in the hyperlist then begin by selecting the last node in the probableSatelliteNodes list and map it. Repeat the process until all nodes in the hyperlist are mapped

Step 9: Place the remaining nodes that are finally left in the probableSatelliteNodes set as satellite nodes to the primary nodes with load balancing taken into consideration. If the satellites are at a physical distance greater than 1, change the order of neighbors with similar node degrees at first encountered entry with this property in the adjacency list. We assume that changing the order will result in different embedding. Go to Step 1.

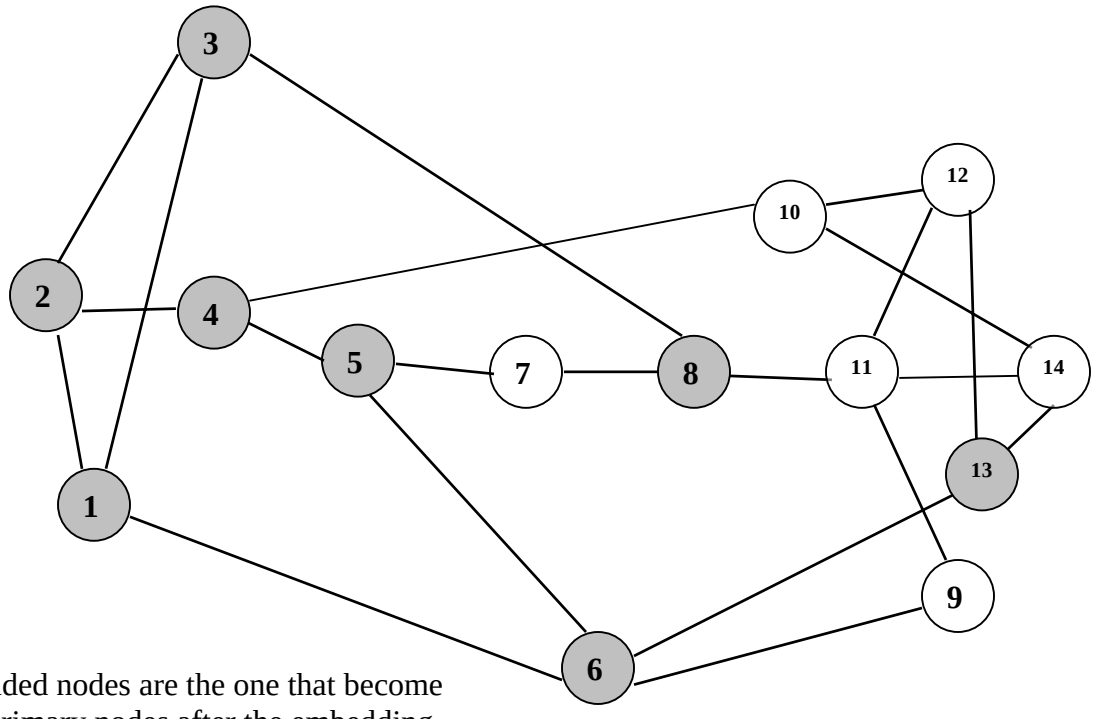
Step 10: If two hypercube neighbors were mapped to neighbors in the physical network, the link connecting them needs no further processing. Such links are colored green. All non-green hypercube links must be processed further by Phase 2 of the algorithm. Place such links in a list and pass it on to Phase 2 of the algorithm.

Step 11: Go to Phase 2.

Once an embedding has been achieved and the satellite nodes have been placed, the second phase aims to find the physical paths for the virtual links in the embedded hypercube. This phase is discussed in Section 3.3.

We explain the steps involved in the first phase of the algorithm through an example. Consider the NSFNet shown in Figure 3.1. Table 3.1 is the adjacency list of the nodes of the NSFNet. Table 3.2 is the sorted adjacency list that was sorted based on the node degrees. Neighbors of each node are listed in non-increasing order of their node degrees. Table 3.3 is the adjacency list of a 3-dimensional hypercube or a hyperlist of order three. We choose the 3-dimensional hyperlist as it satisfies the condition in Step 2 of the algorithm.

Node 6 and node 11 are the nodes with the highest node degree. Hence the node that comes across first in the sorted list (node 6) is assigned to one of the hypercube nodes. We then move to the neighbors of node 6 that have node degrees equal to or are one less than that of node 6. These nodes are 1, 5 and 13 that are assigned as neighbors to node 6 in the hypercube. Node 9, which is also a neighbor to node 6, is placed in the probable satellite list since it has a node degree lower than 6 by more than one. After this first iteration, neighbors of each of the placed nodes, i.e., 1, 5 and 13, are processed. By performing the same process iteratively, we arrive at the embedding shown in Figure 3.2 and an associated list of satellite nodes. The next step of the algorithm is to assign the nodes listed as satellites to the primary nodes with load balancing in mind. Such satellites should be allocated as evenly as possible to the primary nodes with the constraint that no satellite node is at a physical distance of more than 1 from its assigned primary node.



*Shaded nodes are the one that become the primary nodes after the embedding algorithm is executed.

Figure 3.1: NSFNet Architecture

The final embedding achieved is as shown in Figure 3.3. The bold lines in the figure are the links that mapped with a dilation of 1 to the physical topology and need no further processing. The dashed lines are the links which will have dilation greater than 1 and hence will require physical path allocation. This is the job of the second phase of the algorithm that we discuss in Section 3.3.

Table 3.1: Adjacency list of NSFNet

Node Number	Node degree	Neighbors
1	3	2, 3, 6
2	3	1, 3, 4
3	3	1, 2, 8
4	3	2, 5, 10
5	3	4, 6, 7
6	4	1, 5, 9, 13
7	2	5, 8
8	3	3, 7, 11
9	2	6, 11
10	3	4, 12, 14
11	4	8, 9, 12, 14
12	3	10, 11, 13
13	3	6, 12, 14
14	3	10, 11, 13

Table 3.2: Sorted adjacency list of NSFNet

Node Number	Node degree	Neighbors
6	4	1, 5, 9, 13
11	4	8, 12, 14, 9
3	3	1, 2, 8
4	3	2, 5, 10
5	3	4, 6, 7
1	3	6, 3, 2
8	3	3, 7, 11
10	3	4, 12, 14
2	3	1, 3, 4
12	3	11, 10, 13
13	3	6, 12, 14
14	3	11, 10, 13
9	2	6, 11
7	2	5, 8

Table 3.3: 3-dimensional hyperlist

Node Number	Node Neighbors
000	100, 010, 001
001	101, 011, 000
011	111, 001, 010
010	110, 000, 011
100	000, 110, 101
101	001, 111, 100
111	011, 101, 110
110	010, 100, 111

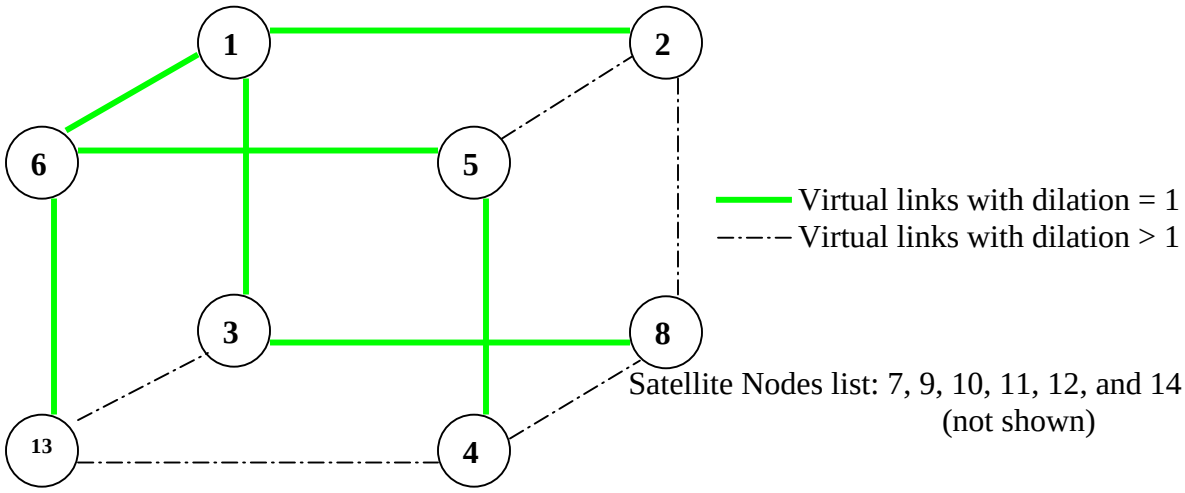


Figure 3.2: Initial embedding of NSFNet

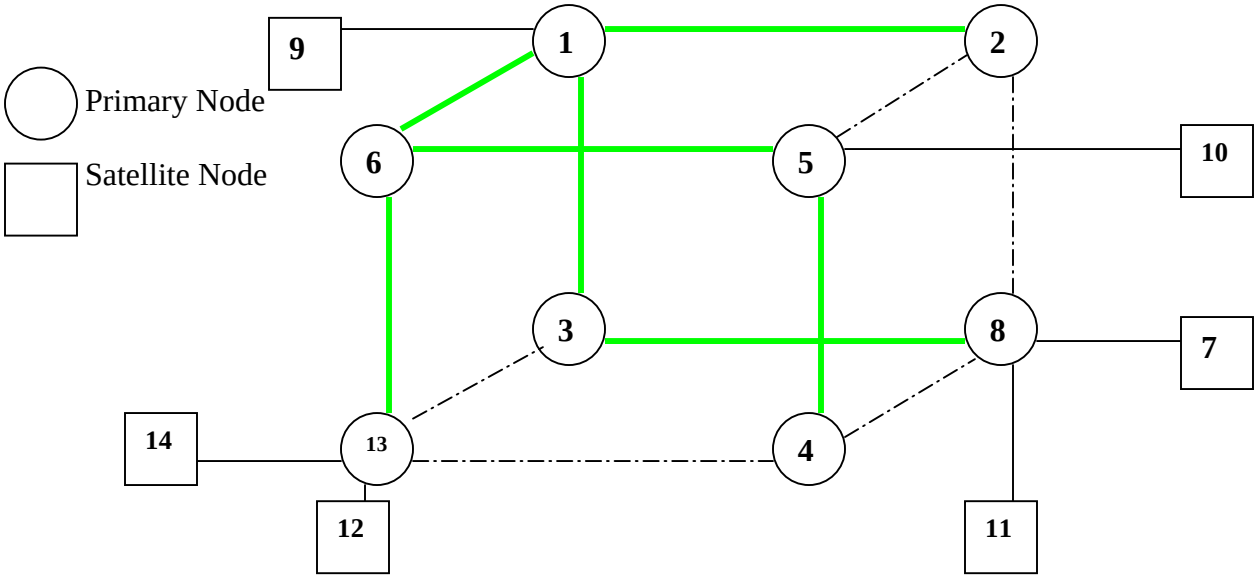


Figure 3.3: Final embedding of a hypercube in NSFNet with satellites assigned to the primary nodes with load balancing considerations.

3.2.1.2 Cycle Method 1 Algorithm

The Cycle Method 1 Algorithm is based on finding a physical cycle in the host graph that has to satisfy two important requirements.

- It must encompass 2^n nodes, where n is the dimensionality of the hypercube chosen.
- It must include the node with the highest node degree.

Such a cycle is called a "primary cycle". To find such a cycle we use a technique based on the depth first search algorithm [30]. When we run this algorithm, we get a set of cycles. We stop the algorithm as soon as we find a cycle that satisfies the above two conditions. Such a cycle is then placed on a Hamiltonian cycle of the virtual hypercube. Most of the time it is not very difficult to find a cycle satisfying the two conditions since most practical networks have reasonably good connectivity. However, if such a cycle is not present then we find a virtual cycle that has 2^n nodes and encompasses the node with the highest node degree. In this case, the number of virtual links that are mapped with a dilation greater than 1 is increased. We assume that such a cycle always exists. While constructing this cycle, we must use as many physical links as possible to increase the number of virtual links that are mapped with a dilation of 1. The remaining nodes which are not part of the primary cycle, whose number is $|P| - |V|$ are placed as satellites to the primary nodes while satisfying the following two conditions.

- Each satellite node is at a maximum physical distance of 1 from the primary node to which it is assigned. (We assume that this is always possible).
- The association (or assignment) of satellite nodes with the primary nodes is balanced as evenly as possible.

The Cycle Method 1 algorithm is stated as follows.

Step 1: Examine the host network, read it into a structure and find the total number of nodes (N_p).

Step 2: Determine the dimensionality of the required hypercube, *n*, which satisfies the following: Find the largest n such that $2^n < |P|$. Once *n* is found we have the set *V* of virtual nodes such that $2^n = |V|$. The set *V* contains the labels {0, 1, 2, … , $2^n - 1$}.

Step 3: Choose the appropriate hyperlist based on *n* found in Step 2.

Step 4: Execute a Depth First Search (DFS) algorithm on the host network. Find the first cycle of the network whose group size is equal to the number of nodes in the hypercube and that encompasses the node with highest node degree. If such a cycle does not exist, go to Step 6.

Step 5: Place this cycle on the known Hamiltonian cycle of a hypercube of dimension *n*. Go to Step 7.

Step 6: Find a virtual cycle that encompasses the node with the highest node degree and place this cycle on the Hamiltonian cycle of the hypercube.

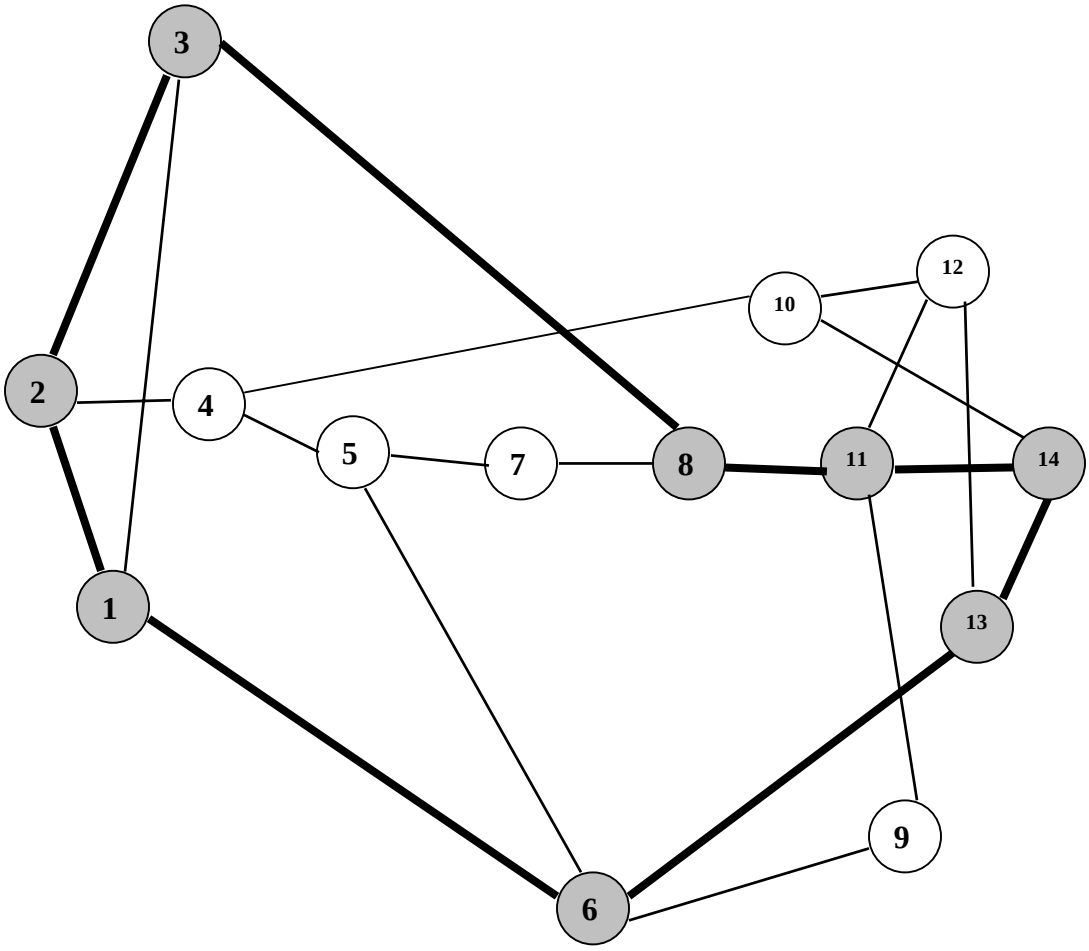
Step 7: Assign the remaining nodes, which are not part of the primary cycle, in the probableSatelliteNodes list as satellite nodes to the Primary nodes while considering load balancing.

Step 8: If two hypercube neighbors were mapped to neighbors in the physical network, the link connecting them needs no further processing. Such links are colored green. All non-green hypercube links must be processed further by Phase 2 of the algorithm. Place such links in a list and pass it on to Phase 2 of the algorithm.

Step 9: Go to Phase 2.

Once an embedding is achieved and all the satellite nodes are placed, the second phase of the algorithm is executed to find the physical path implementation for the virtual links. This phase is discussed in Section 3.3.

To illustrate the algorithm, again consider the NSFNet, with 14 nodes and 20 links, shown in Figure 3.4.



- Bold lines indicate the chosen cycle
- The shaded nodes form the primary nodes

Figure 3.4: NSFNet architecture with the primary cycle shown.

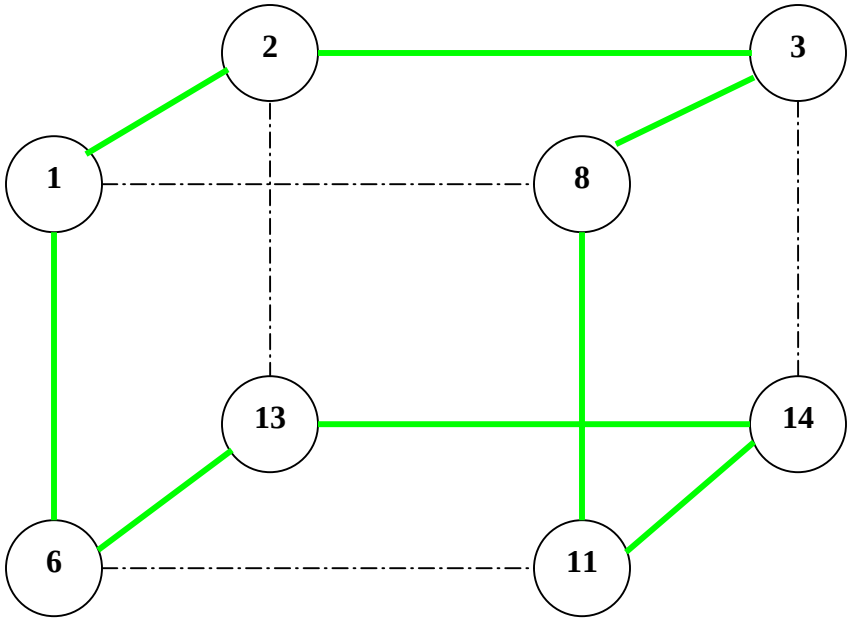


Figure 3.5: 3-dimensional hypercube with a Hamiltonian cycle shown in bold lines. This cycle also represents the embedded primary cycle of the physical topology shown in Figure 3.4.

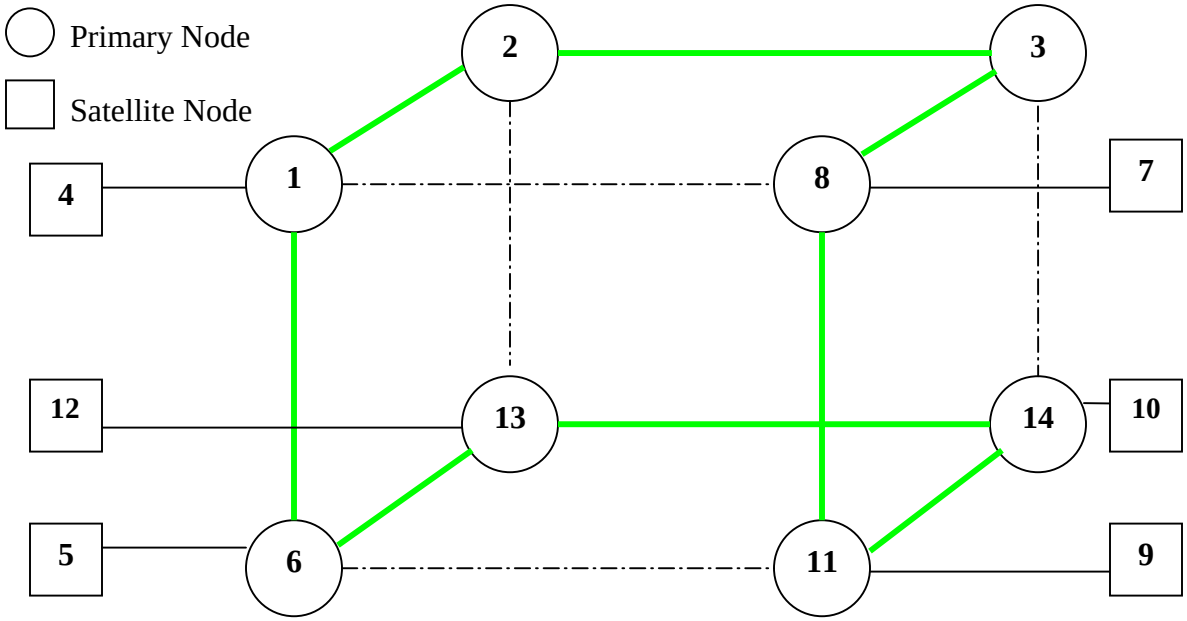


Figure 3.6: Final embedding into the NSFNet using Cycle Method 1.

Once the network is analyzed by examining its adjacency list as in Table 3.1 and the sorted adjacency list as in Table 3.2, we choose the appropriate hyperlist, which is for a 3-dimensional hypercube. This is shown in Table 3.3. Executing the DFS algorithm can find all the cycles in the network. We interrupt the algorithm once we find a cycle that satisfies the two conditions listed earlier. This chosen cycle is called the "primary cycle" and is shown in Figure 3.4 in bold lines within the NSFNet architecture. This cycle is placed on one of the known Hamiltonian cycles of the hypercube as shown in Figure 3.5. The next step is to assign the satellite nodes to primary nodes so that satellites are as evenly distributed as much as possible but with the constraint that no satellite node is at a physical distance of more than 1 from its associated primary node. The final embedding achieved is as shown in Figure 3.6. The bold lines in the figure are the links that mapped with dilation one to the physical topology and need no further processing. The dashed lines are the links that will have dilation greater than one, and hence will require physical path assignment. This is the job of the second phase of the algorithm, which we discuss in Section 3.3.

3.2.1.3 Cycle Method 2 Algorithm

The Cycle Method 2 Algorithm is a variant of Cycle Method 1 algorithm. It is based on finding a physical cycle in the host graph that has to satisfy two slightly different requirements.

- It must encompass 2^n nodes, where n is the dimensionality of the hypercube chosen.
- It must include nodes with higher than average node degree (rather than just the node with the highest node degree). A node with higher than average node degree is found by finding the average node degree of the network and then selecting nodes that have a node degree greater than the average to be included in the cycle.

Such a cycle is called a “primary cycle”. We use the same technique to find this cycle as for Cycle Method 1, i.e., based on the depth first search algorithm [30]. When we run this we could get a set of cycles. However we stop the algorithm as soon as we find a cycle that satisfies the above two conditions. Such a cycle is then placed on a Hamiltonian cycle of the virtual hypercube. Most of the time it is not very difficult to find a cycle satisfying the two conditions since most practical networks have good connectivity and node degrees do not differ very significantly. However, if such a cycle is not present then we find a virtual cycle that has 2^n nodes and encompasses the nodes with node degrees greater than the average node degree of the network. We assume that such cycle always exists. In this case, the number of virtual links that have dilation greater than 1 is increased. Such a virtual cycle is constructed utilizing nodes from the set of nodes having a higher than average node degree. While constructing this cycle, we must use as many physical links as possible to increase the number of virtual links that are mapped with a dilation of 1. The remaining nodes that are not part of the primary cycle, i.e., is $\{P\} - \{V\}$, are placed as satellites to the primary nodes while satisfying the following two conditions.

- Each satellite node is at a maximum physical distance of 1 from the primary node to which it is assigned. (We assume that this is always possible).
- The association (or assignment) of satellite nodes with the primary nodes is balanced as evenly as possible.

The Cycle Method 2 algorithm is stated as follows.

Step 1: Examine the host network, read it into a structure and find the total number of nodes (N_p), find the average node degree of the network.

Step 2: Determine the dimensionality of the required hypercube required n that satisfies the following. Find the largest n such that $2^n < |P|$. Once n is found we have the set V of virtual nodes such that $2^n = |V|$; $V = \{0, 1, 2, \ldots 2^n\text{-}1\}$.

Step 3: Choose the appropriate hyperlist based on n found in Step 2.

Step 4: Execute a Depth First Search (DFS) algorithm on the host network. Find the first cycle of the network whose group size is equal to the number of nodes in the hypercube and that encompasses the nodes with node degree greater than the average node degree of the network if it exists. If such a cycle does not exist, go to Step 6.

Step 5: Place this cycle on the known Hamiltonian cycle of the hypercube. Go to Step 7.

Step 6: Find a virtual cycle that encompasses the nodes with greater than average node degree and place this cycle on the Hamiltonian cycle of the hypercube.

Step 7: Place the remaining nodes, which are not part of the primary cycle, in the probableSatelliteNodes list as satellite nodes to the primary nodes while considering load balancing.

Step 8: If two hypercube neighbors were mapped to neighbors in the physical network, the link connecting them needs no further processing. Such links are colored green. All non-green hypercube links must be processed further by Phase 2 of the algorithm. Place such links in a list and pass it on to Phase 2 of the algorithm.

Step 9: Go to Phase 2.

To understand the algorithm better, we explain the flow of the algorithm with an example. Consider the network known as vBNS (very high-speed Backbone Network Service) with 15 nodes and 20 links shown in Figure 3.7. A hypercube of dimension 3 satisfies the condition in Step 2. The DFS algorithm finds all the cycles in the physical topology. However, we restrict the search algorithm to finding the cycles that satisfy the conditions of the Cycle Method 2 stated in Step 4. The cycle found by DFS is shown in Figure 3.7 in bold lines. This cycle is placed on the Hamiltonian cycle of the 3-dimensional hypercube as shown in Figure 3.8. . The next step is to assign the satellite nodes to primary nodes so that satellites are as evenly distributed as much as possible but with the constraint that no satellite node is at a physical distance of more than 1.

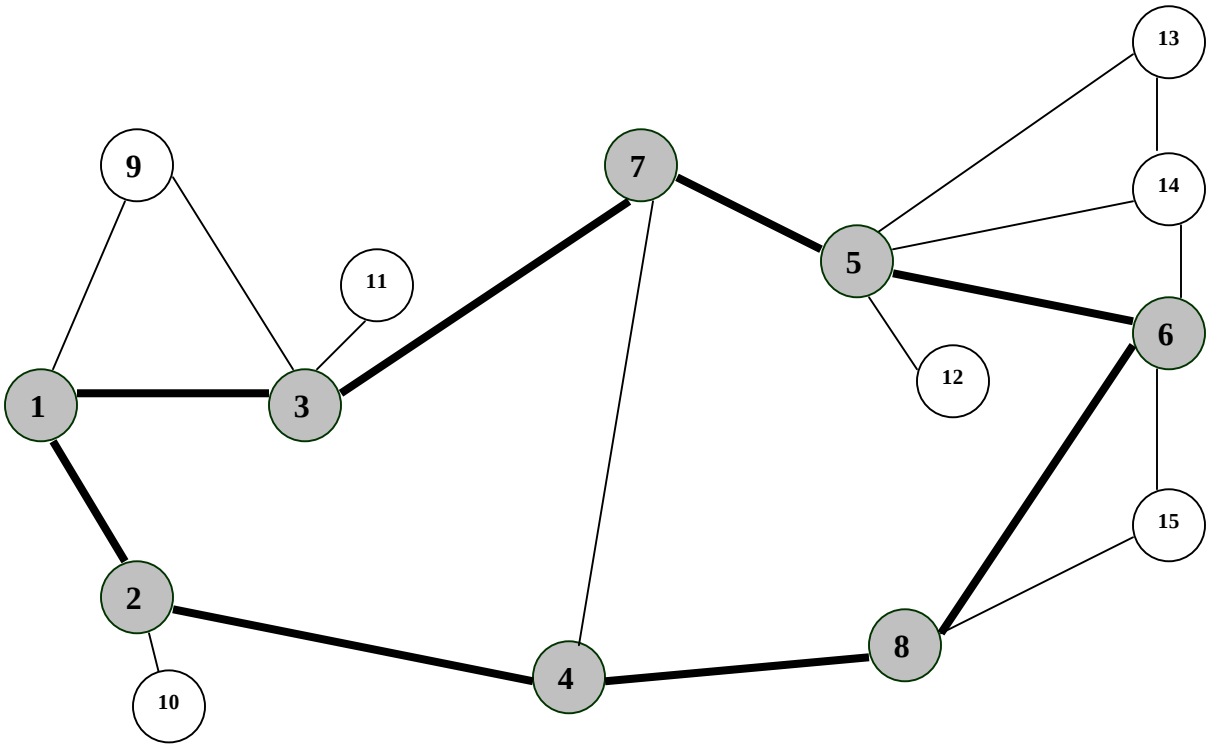


Figure 3.7: vBNS network architecture

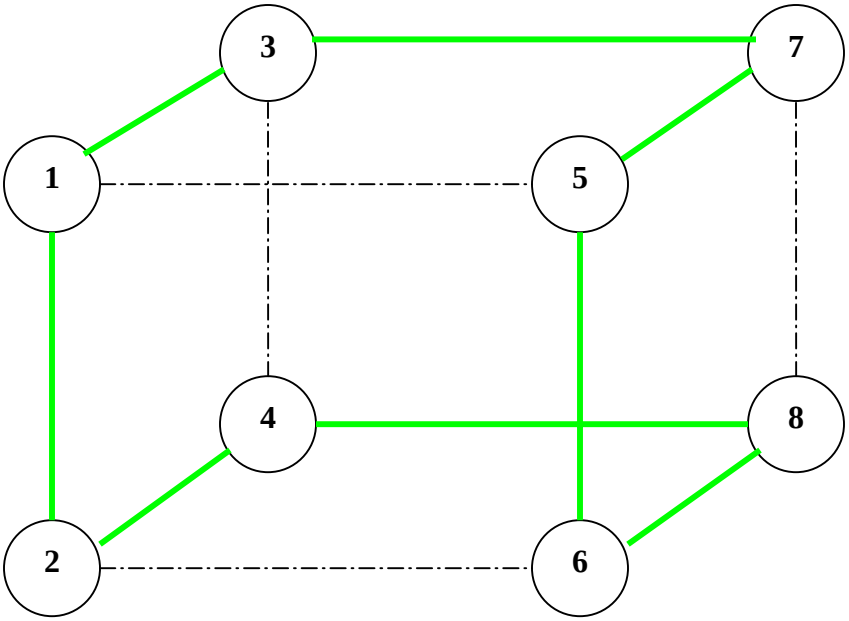


Figure 3.8: 3-dimensional hypercube with Hamiltonian cycle shown in bold lines. This cycle also represents the embedded primary cycle of the physical topology shown in Figure 3.7.

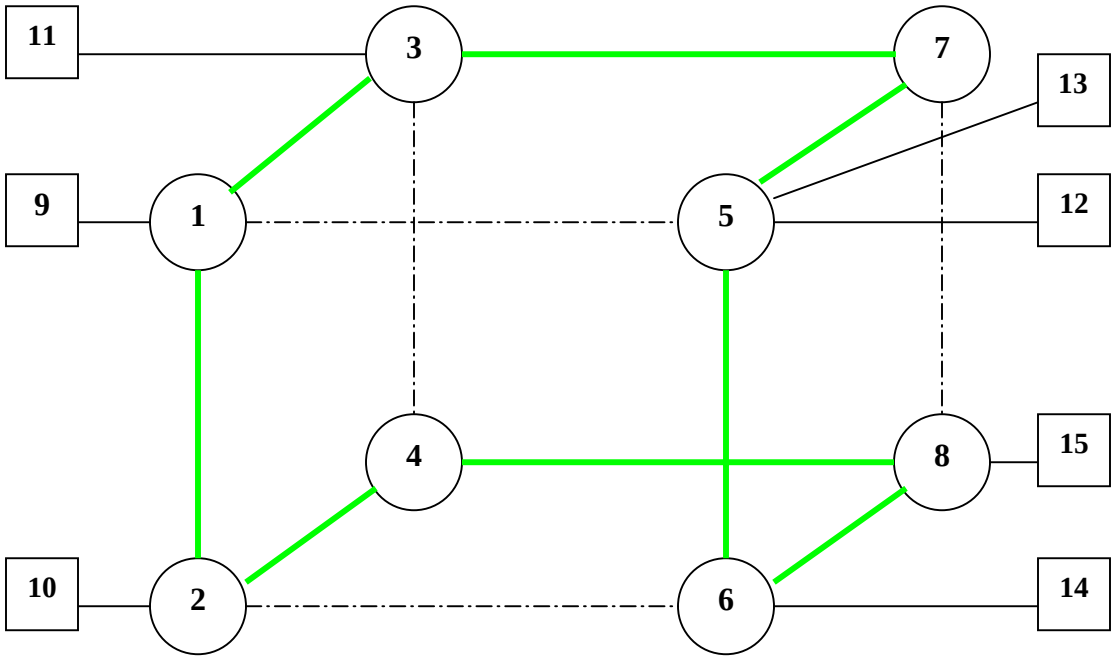


Figure 3.9: Final embedding into the vBNS network using Cycle Method 2.

The final embedding achieved is as shown in Figure 3.6. The bold lines in the figure are the links that map with a dilation of 1 to the physical topology and need no further processing. The dashed lines are the links which will map with dilation of greater than 1 and hence will require physical lightpath assignment. Once a node mapping is achieved, we move to the second phase of the algorithm, which performs virtual path implementation. This phase is discussed in the following section.

3.2.2 Mapping the Virtual Connections

Once the nodes have been assigned virtual addresses and the satellite nodes assigned to the appropriate primary nodes, a path through the physical network must be found for each connection between nodes that are adjacent in the virtual topology. This step of the algorithm attempts to find a physical path for each virtual connection so that the number of wavelengths on each physical link is minimized. In this step each virtual link is realized by a path that could consist of a series of physical edges. The length of a virtual connection is the sum of the lengths of the physical edges that connect the virtually adjacent nodes. To reduce the number of WDM channels required on each physical link, this phase of the algorithm seeks to minimize the maximum number of virtual connections that traverse any particular physical link. The maximum number of virtual connections traversing a physical link provides a lower bound on the number of wavelengths required to provide the optical connections. The algorithm must attempt to select among paths, which are near shortest path but are as disjoint from each other as possible in order to minimize the number of wavelengths.

Several routing algorithms have been designed in the literature to find virtual path implementations [19, 25, 31]. Most of the methods assume the traffic load to be a continuous

function that can be divided among an arbitrary number of paths. Unfortunately these algorithms are intractable when dealing with the case of allocating entire individual circuits. Therefore we have developed a simple heuristic algorithm for balanced routing based on the concept of finding disjoint shortest paths, whenever possible. When not possible, we try to find disjoint non-shortest paths. When both types of the above mentioned paths are not possible to find, we implement such virtual circuits with their shortest paths. Avoiding iteration and making the allocations in a single pass over the circuits in sorted order tends to simplify the algorithm. A virtual link with dilation 1 embedding does not need any more processing. Virtual links that have dilation greater than 1 are listed in a table and all their possible shortest physical paths are tagged along with them. The virtual connections whose shortest paths are disjoint are implemented first. For implementing virtual links whose shortest paths are not disjoint we try to find disjoint "non-shortest" paths.

The algorithm to find disjoint paths such that the number of wavelengths is reduced is described in the following steps.

- Make a list of all node pairs that require virtual connections to complete the hypercube and let A { } be the set of all such node pairs. These are essentially the list of non-green hypercube links passed by the first phase of any of the three algorithms.
- List all the shortest paths possible between each of these node pairs in a table.
- Select all the disjoint shortest paths available in this table and allocate those physical paths for the virtual connections between the corresponding node pairs.
- Form a set U { } that contains all individual hops that make up the already selected physical paths. Initially, U { } is an empty set.

- Form another set M { } that contains all the node pairs (of adjacent virtual nodes) that have been assigned physical paths for their virtual connections.
- Let R { } = {A – M} be the set of remaining node pairs. For each remaining node pair in R { } try finding a physical path through the original network using hops not listed in U. This path need not be a shortest path. If finding such a path is not possible, then use a shortest path by utilizing the links listed in U but make an entry of that hop in another set D { }.

The above procedure attempts minimization of the number of wavelengths required for routing in the hypercube. For a given embedding solution, the number of wavelengths required for embedding an n-dimensional virtual hypercube is equal to V`, where V` is the maximum number of virtual links mapped to the same physical link. The lower bound would be equal to 1 as that particular physical link is either mapped as a virtual link with dilation 1 or is one of the links being mapped to form the virtual link between neighbors of the hypercube.

Let us now apply this second phase of the algorithm to one of the embedding we developed earlier. Consider the embedding achieved using the direct method shown in Figure 3.3. We first define the set A and list all the shortest paths for each of the virtual links that require further processing as shown in Table 3.4 below.

Table 3.4: List of to be processed virtual links with their shortest paths

A { } (List of virtual links that have a dilation > 1 or the links which need further processing)	List of all possible shortest paths
5-2	5-4-2
8-2	8-3-2
13-4	13-12-10-4, 13-14-10-4, 13-6-5-4
13-3	13-6-1-3
4-8	4-5-7-8

By selecting all disjoint shortest paths, we can find physical path implementation for four out of the five node pairs that require processing. These node pairs form the elements of the set M { }. The links that have already been used to make these physical connections are updated in the set U { }. The values in the set M { } and set U { } are as follows:

M = {(5-2), (8-2), (13-4), (13-3)}

U = {(2-3), (3-8), (5-4), (4-2), (13-12), (12-10), (10-4), (13-6), (6-1), (1-3)}

The set R { } consists of the list of node pairs that still need physical path implementations and is given by R { } = A – M. Hence in this case R = {(4-8)}. We have to find a path between the nodes 4 and 8 such that they do not use any of the links listed in the set U { } if possible. This path need not be a shortest path. In this case we cannot find such a path for (4-8). Hence we implement the shortest path and make an entry in the set D { } of the links that are already present in U { }. Hence the set D = {(4-5)}.

The results and analysis of the above-discussed heuristic algorithms will be presented in the following chapter.

Chapter 4

Implementation and Results

In this chapter we present the software implementation and the performance metrics for the heuristic algorithms discussed in the previous chapter. We compare the behavior of all three algorithms and show which algorithm works best under different conditions. We ran these algorithms on various practical and randomly generated experimental networks with different connectivities and compared their results. To arrive at the results we designed software to develop the embeddings according to our algorithms and to analyze the embeddings with respect to certain parameters such as average virtual path length, average number of wavelengths per link, etc.

4.1 Software Implementation

To validate any heuristic algorithm, we need a way to test its performance. In order to test our algorithms, we designed software that would run on any irregular network and produce the embeddings following the steps detailed in Chapter 3. The software was developed in the C language. It accepts the adjacency list of any network. The adjacency list as defined in this thesis corresponds to a 3-column structure in which the first column contains the node number, the second contains the node degree and the third is an array which consists of the neighbors of the node (the number of nodes listed in the third column as neighbors is always equal to the node degree listed in the second column for a particular first column node). The depth of the adjacency list equals the number of nodes in the network. After reading the adjacency matrix of the input network, we sort the list of neighbors of each node according to their node degree. Then the complete list is sorted according the node degree of the nodes in the first column in descending order. Depending on the number of nodes in the network, a particular degree for the

hypercube is chosen (see Chapter 3). The corresponding hyperlist (adjacency list of the hypercube) is then loaded. After these initial steps, each of the algorithms follows its own set of rules in assigning each node in the hyperlist a particular node in the host network. The output of the software is the adjacency list of the embedded virtual hypercube and a list of satellite nodes. The satellite nodes are then assigned to primary nodes in a balanced manner. After obtaining the embedding, the second phase (of any of the algorithms) is run to find the virtual link implementation using the shortest-disjoint path as much as possible, as described in the previous chapter.

4.2 Comparative Results of the Proposed Algorithms

In this section, we compare the results of the embeddings obtained using all three algorithms. We define certain parameters, which are used to compare the quality of the embeddings achieved. The parameters used are:

- Average physical path length between all virtual node pairs
- Average virtual link length realization
- Average number of wavelengths per link
- Total number of wavelengths needed

4.2.1 Average Physical Path Length between All Virtual Node Pairs

This parameter is defined as the average physical distance between all source-destination pairs in the virtual topology. Once an embedding is achieved we have a certain set of physical links that are used to map the virtual paths in the hypercube for control plane information exchange. There could be certain physical links that are not utilized in the control plane but can be used in the data network. We compute the average physical path length between all node pairs within the virtual hypercube using only the physical links used in the embedding. Hence, this

parameter gives us a way to compare the three algorithms with respect to the quality of virtual path implementation. We evaluate the three algorithms on networks of different sizes, i.e., different numbers of nodes and with variable connectivity. The results are plotted as shown in Figures 4.1-4.5 for 10-node, 15-node, 20-node, 25-node and 30 node arbitrary networks. For every network, we increase the connectivity in increments of 25% of the original connectivity and compute the mean physical path length.

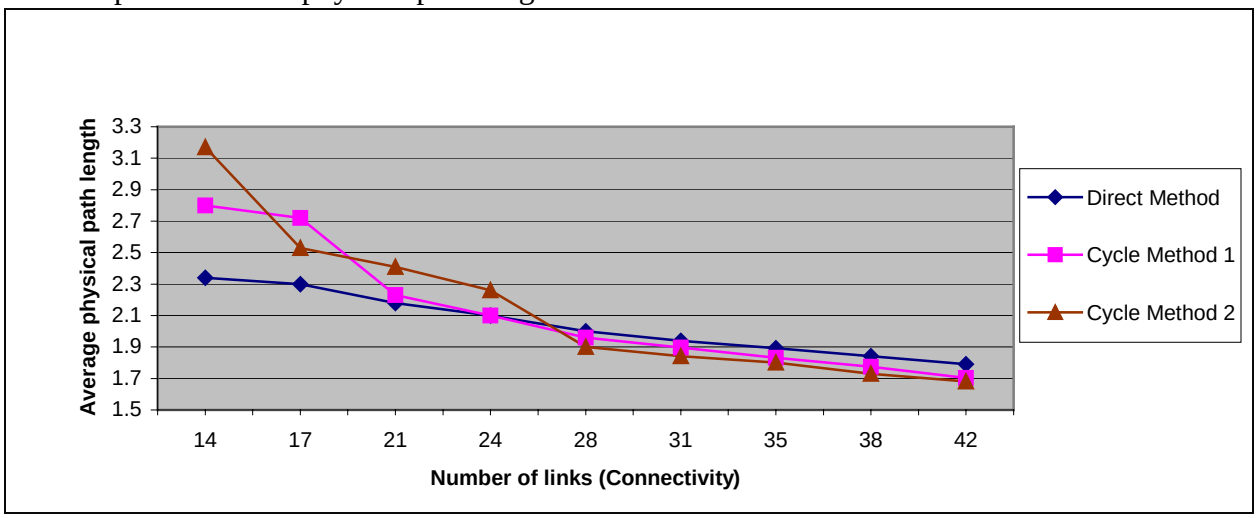


Figure 4.1: Average physical path length between all virtual node pairs for 10-node network with variable connectivities.

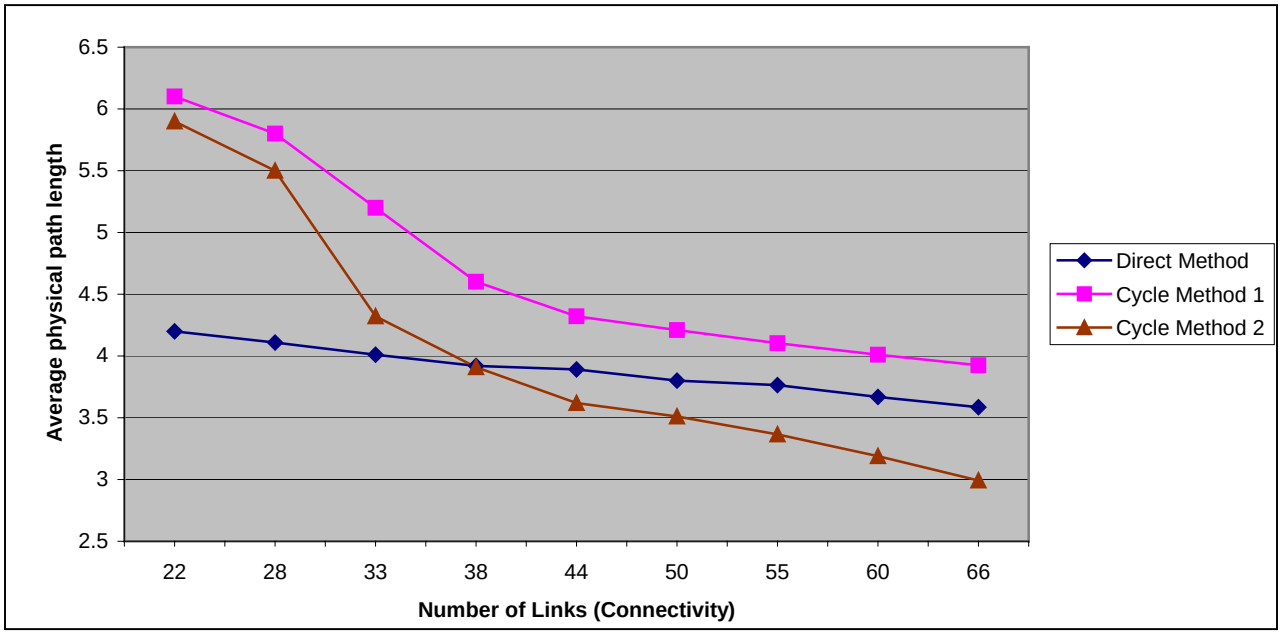


Figure 4.2: Average physical path length between all virtual node pair for 15-node network with variable connectivities.

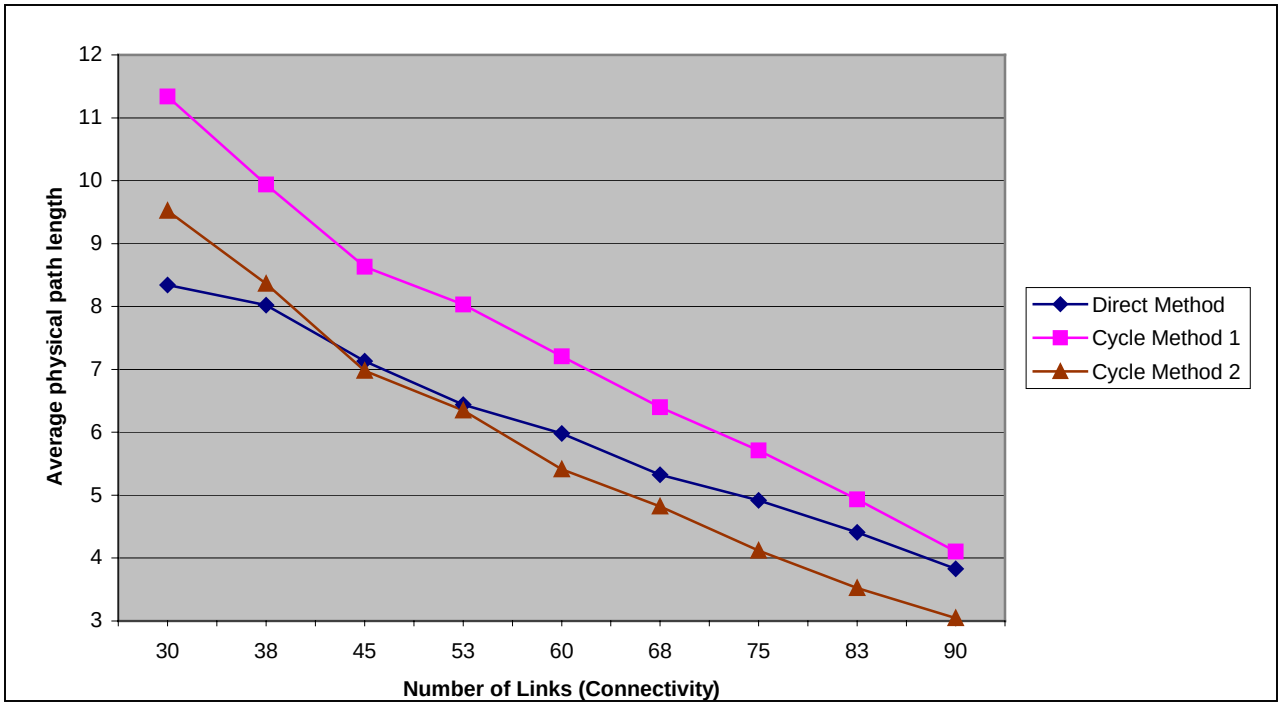


Figure 4.3: Average physical path length between all virtual node pairs for 20-node network with variable connectivities.

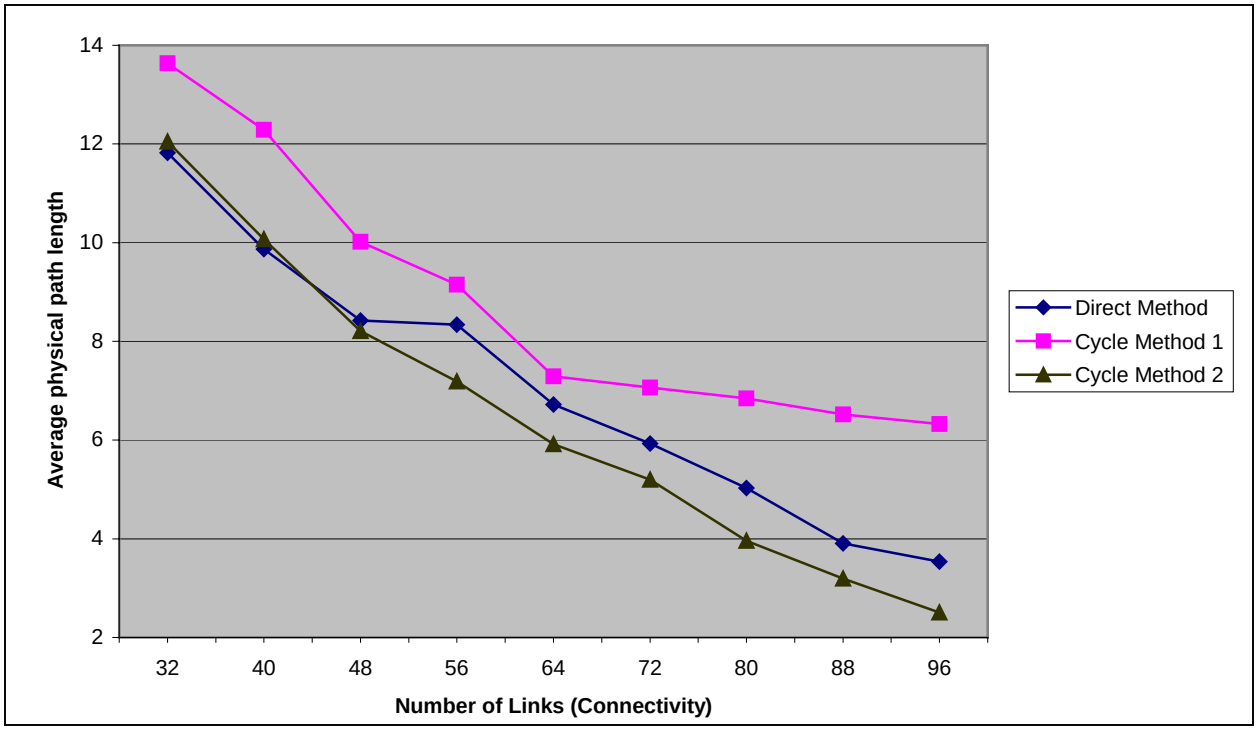


Figure 4.4: Average physical path length between all virtual node pairs for 25-node network with variable connectivities.

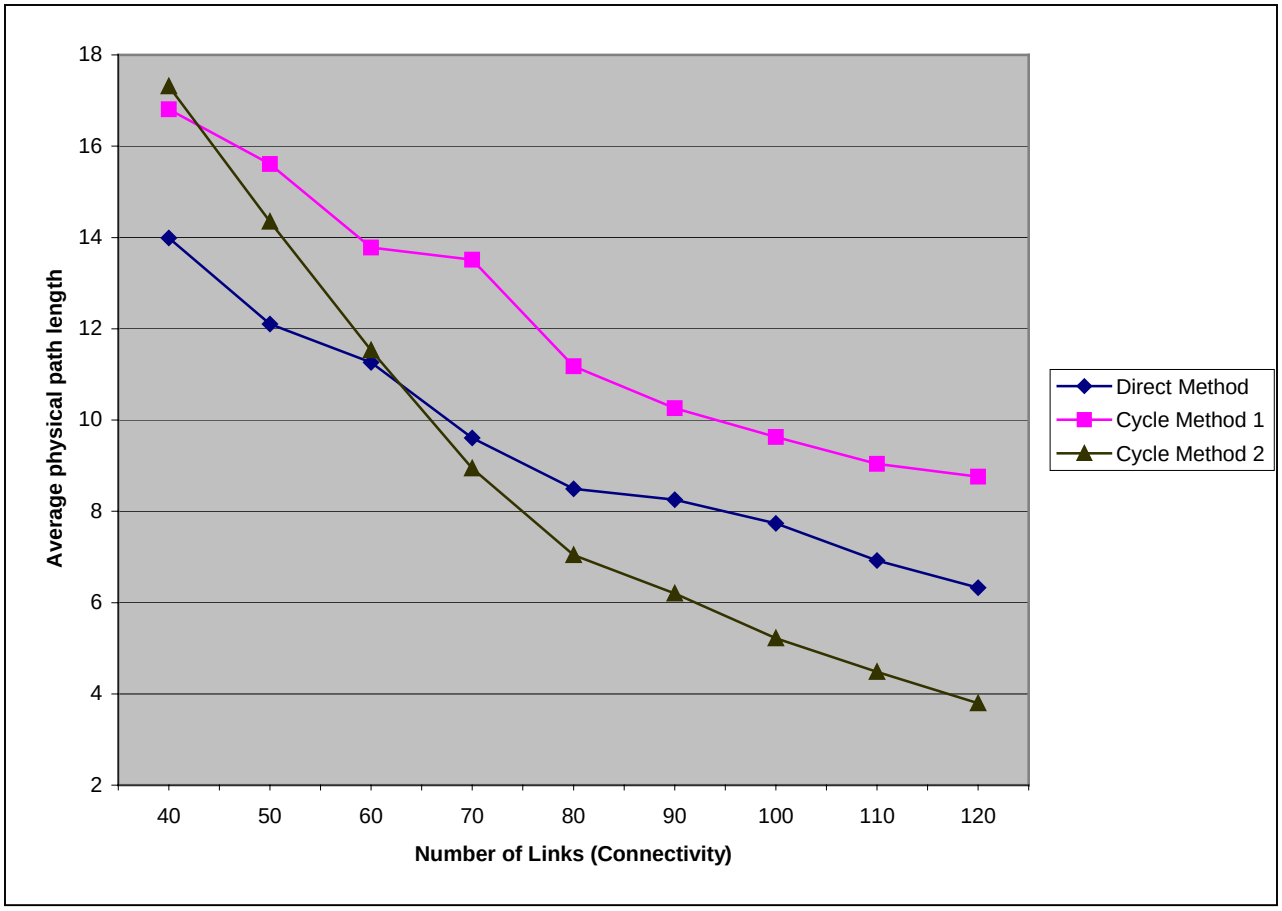


Figure 4.5: Average physical path length between all virtual node pairs for 30-node network with variable connectivities.

As seen in the results in the graphs above, the Direct Method works best for networks of all sizes with smaller connectivity. As connectivity increases, the performance of the other two methods, i.e., Cycle Method 1 and Cycle Method 2, tends to improve. On the whole, we can draw the conclusion that the Cycle Method 2 works the best for networks with higher connectivity and the direct method works best for networks with lower connectivity. On the other hand, Cycle Method 1 works as well as Cycle Method 2 for networks with lower connectivity but is not as good in networks with higher connectivity. The reason being, with an increase in connectivity, instead of concentrating on the node with the highest node degree, Cycle Method 2 includes nodes with above average node degree. This means that more such cycles can be found and better embeddings can be developed.

4.2.2 Average Virtual Link Length Realization

This parameter is defined as the average physical path length realization of the virtual links between hypercube neighbors. If the neighboring nodes in the virtual hypercube are neighbors in the physical topology, then such links are said to have been mapped with a dilation of 1 and the physical path length for such links is taken to be one. If neighboring nodes in the hypercube are not mapped to physically adjacent nodes in the host graph, such neighbors have to be connected using more than one physical link. The dilation in such cases is greater than one. Here we measure the average path lengths of the lightpaths realizing virtual hypercube links. This parameter measures the quality of the embedding with respect to virtual link embedding dilation. Figures 4.6-4.10 show the results for 10-node, 15-node, 20-node, 25-node and 30 node arbitrary networks with variable connectivities.

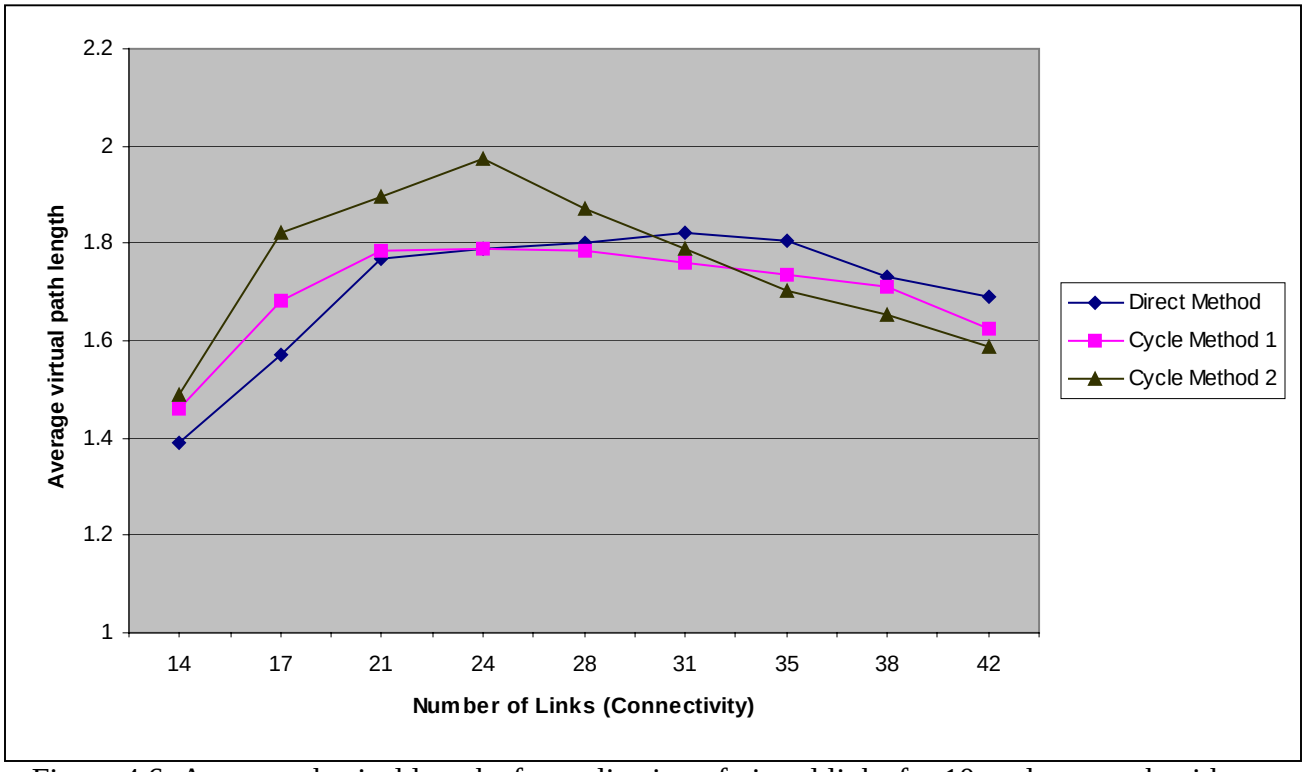


Figure 4.6: Average physical lengths for realization of virtual links for 10-node network with variable connectivities.

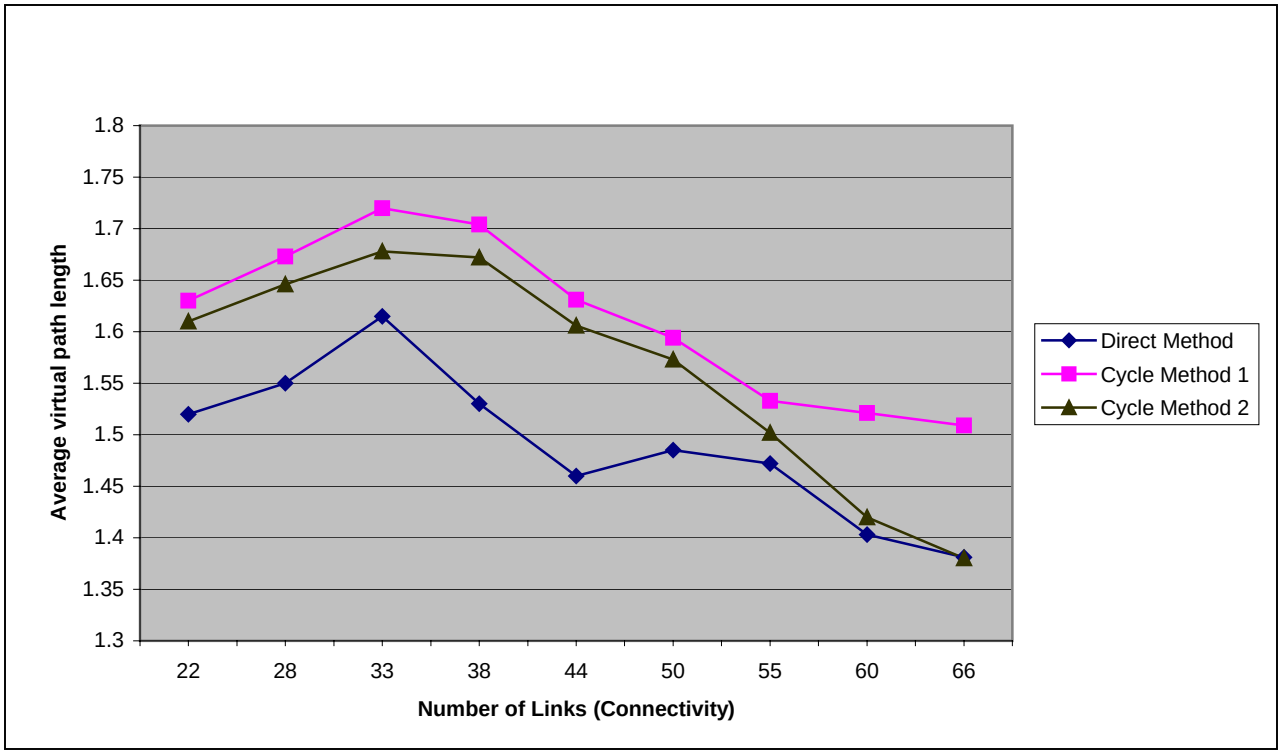


Figure 4.7: Average physical lengths for realization of virtual links for 15-node network with variable connectivities.

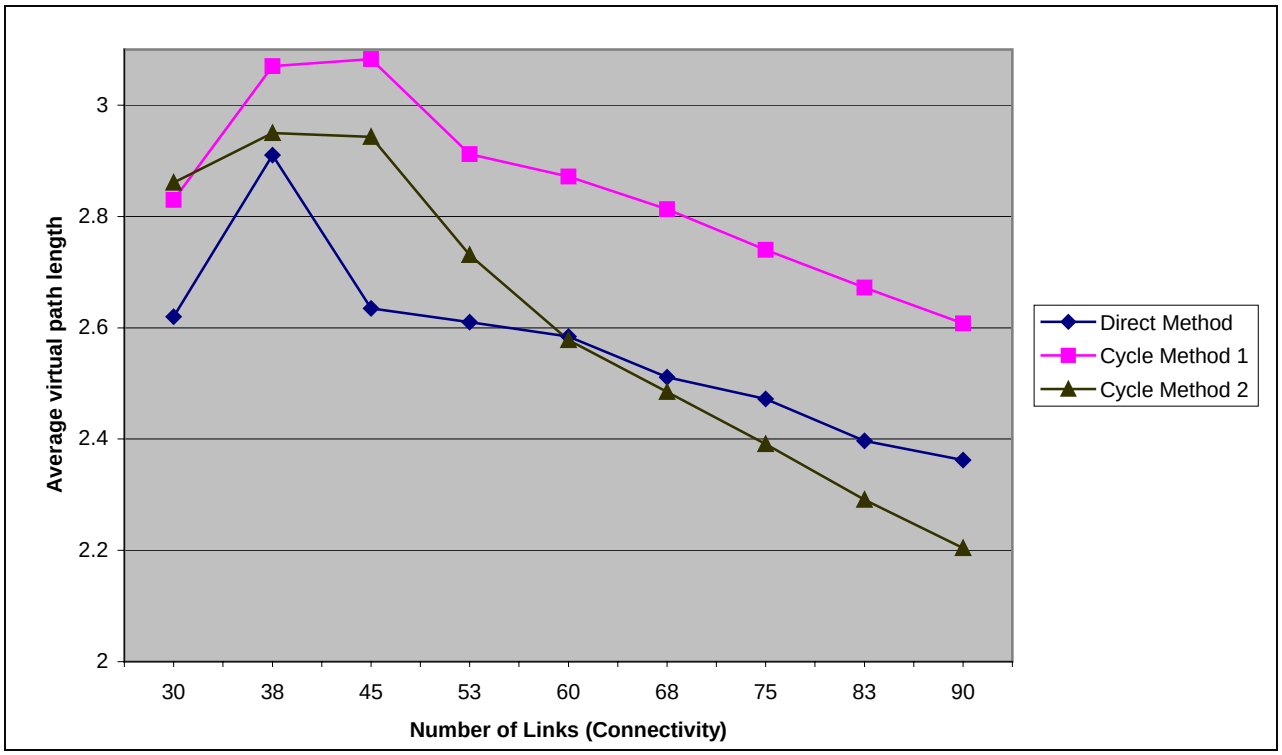


Figure 4.8: Average physical lengths for realization of virtual links for 20-node network with variable connectivities.

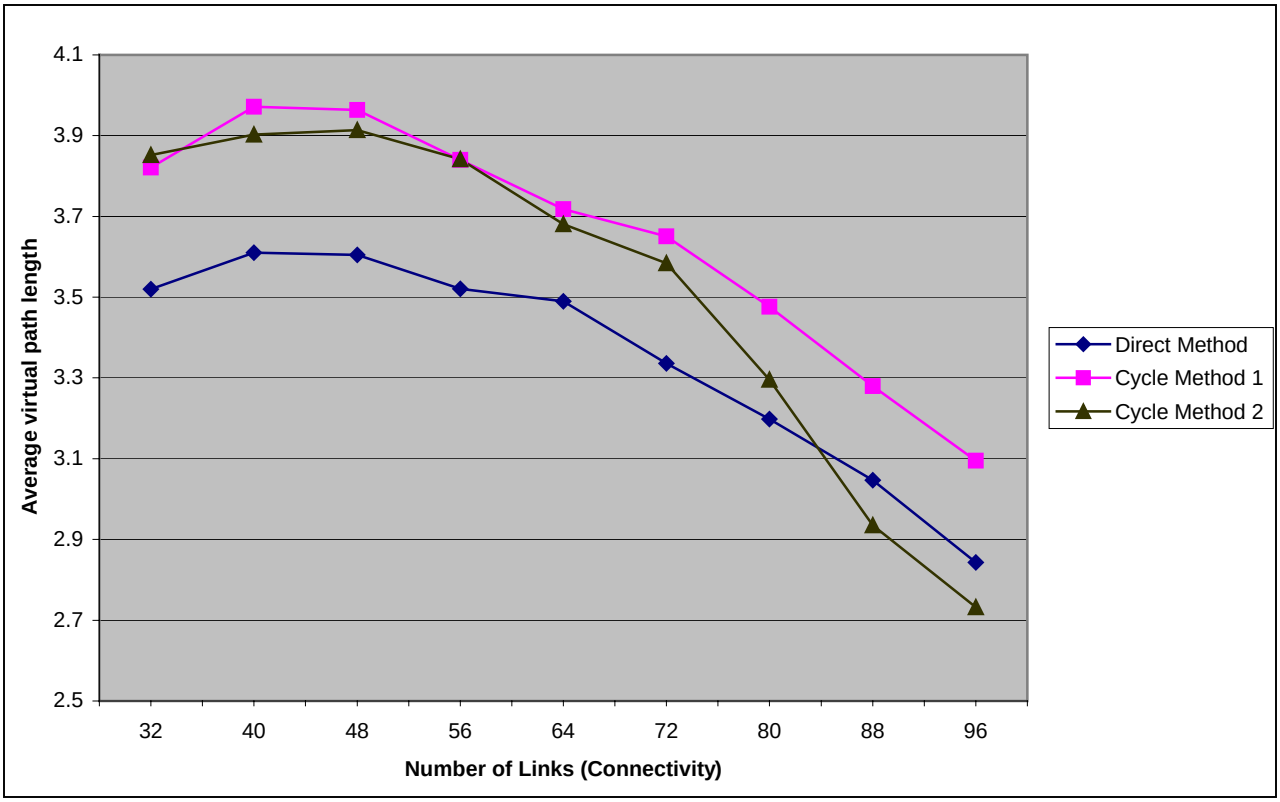


Figure 4.9: Average physical lengths for realization of virtual links for 25-node network with variable connectivities.

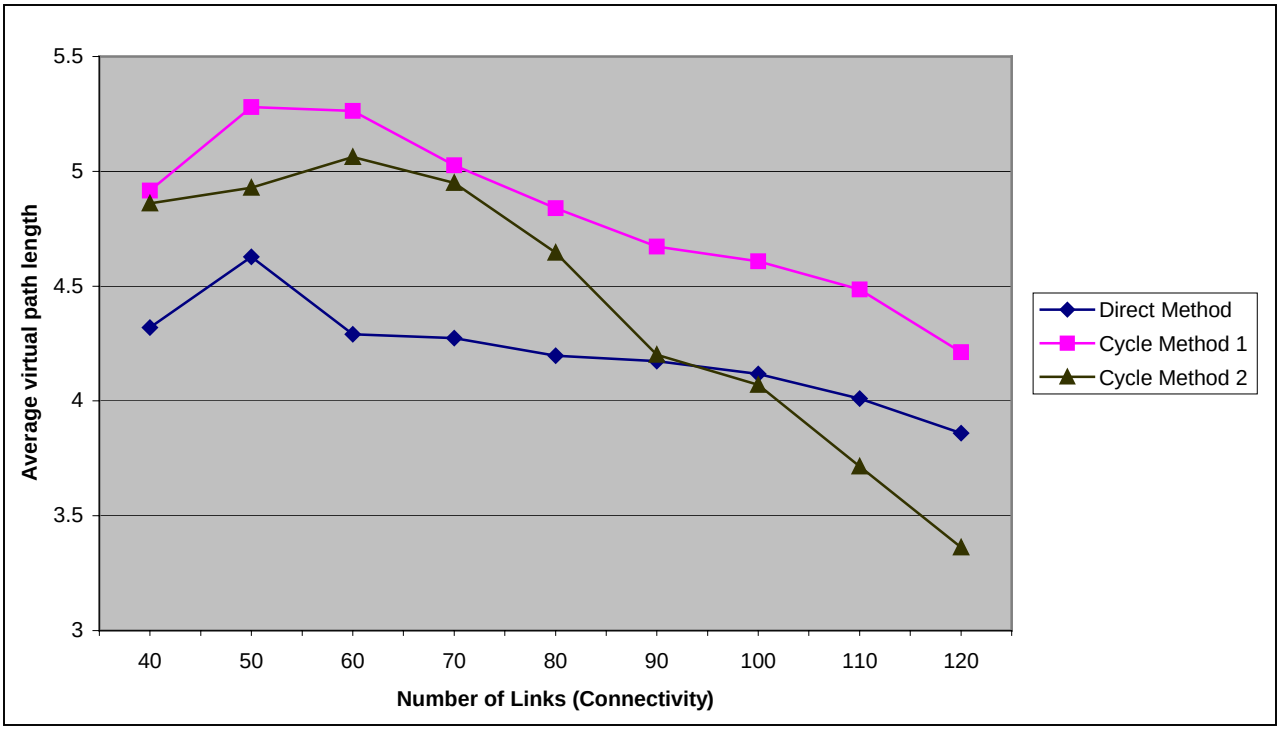


Figure 4.10: Average physical lengths for realization of virtual links for 30-node network with variable connectivities.

As can be observed from the graphs, we see that there is an increase in the average virtual link dilation when we slightly increase connectivity. This is because with some extra links we can find disjoint paths, which are somewhat longer than those found for a network with lower connectivity. Finding disjoint paths tends to reduce the number of wavelengths utilized in the network which is an objective. So, as connectivity increases slightly we expect longer virtual link realization but expect to use fewer wavelengths for the embedding (to be verified in Section 4.2.4). But as we increase the connectivity further, we are able to find shorter paths, which are also disjoint. Hence the average virtual link realization reduces in length after the initial surge. Furthermore, it can be observed that for all these networks, the Direct Method works better. However, as networks become very well connected, then Cycle Method 2 behaves far better than the other two algorithms, producing shorter average virtual link realization.

4.2.3 Average Number of Wavelengths per Link

This parameter is defined as the average number of wavelengths per physical link used to implement the embedding. To obtain it we, add the number of wavelengths on each physical link used and divide by the total number of links used in the embedding. In all our embedding algorithms, we always try to minimize the number of wavelengths by finding disjoint paths. Figures 4.11 - 4.15 show the average number of wavelengths for 10-node, 15-node, 20-node, 25-node and 30-node arbitrary networks with variable connectivity. As can be observed, for smaller networks, the Direct Method tends to give better results. As we increase the connectivity for smaller networks, the Cycle Methods produce comparable results to those for the Direct Method. For larger networks with smaller connectivity, the Direct Method still performs better than the Cycle Methods. However, when the connectivity is increased beyond a certain point, Cycle Method 2 gives a lower average number of wavelengths per link compared to the Direct Method. Cycle Method 1 is clearly not the best choice for bigger network with higher connectivity. This

is because that method concentrates on the node with the highest node degree and most of the time it fails to take advantage of higher connectivity.

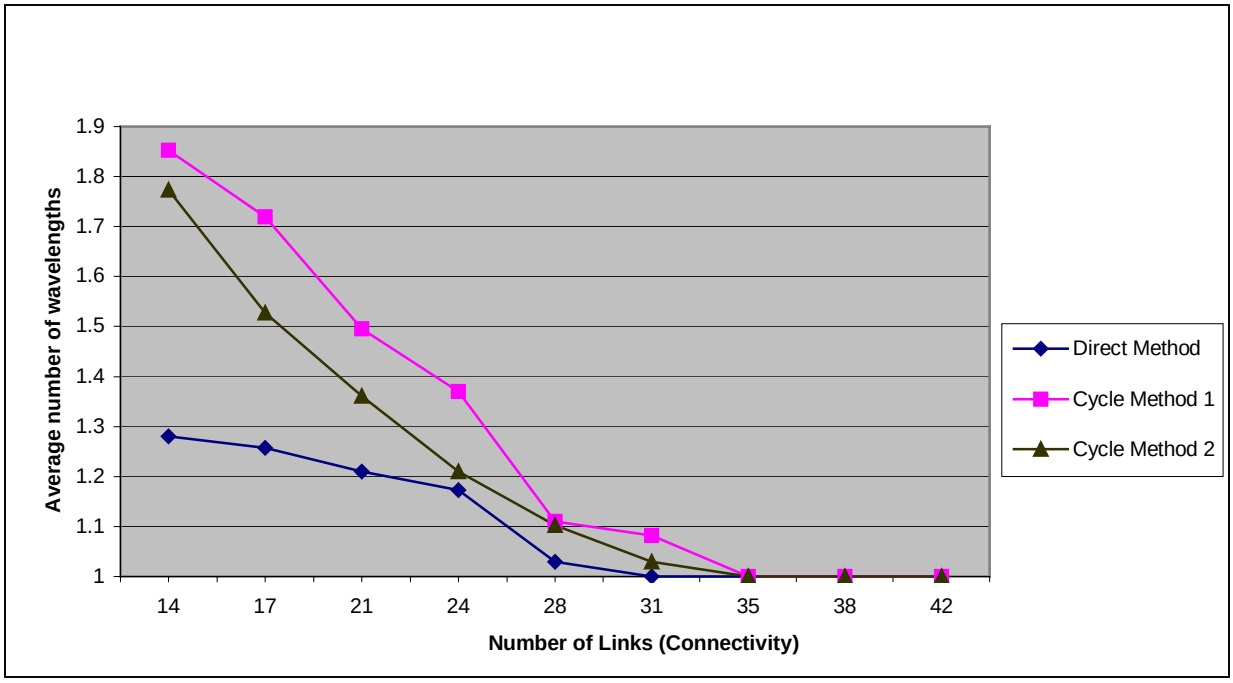


Figure 4.11: Average number of wavelengths on a link for 10-node network with variable connectivities.

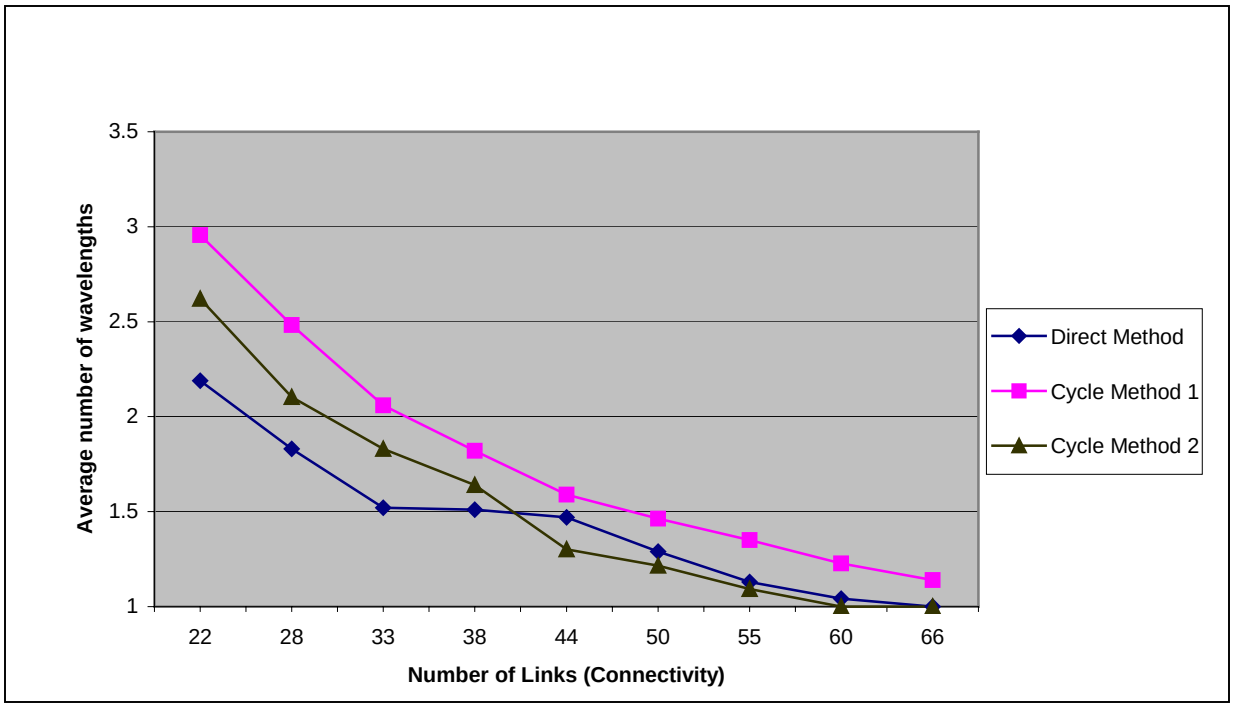


Figure 4.12: Average number of wavelengths on a link for 15-node network with variable connectivities.

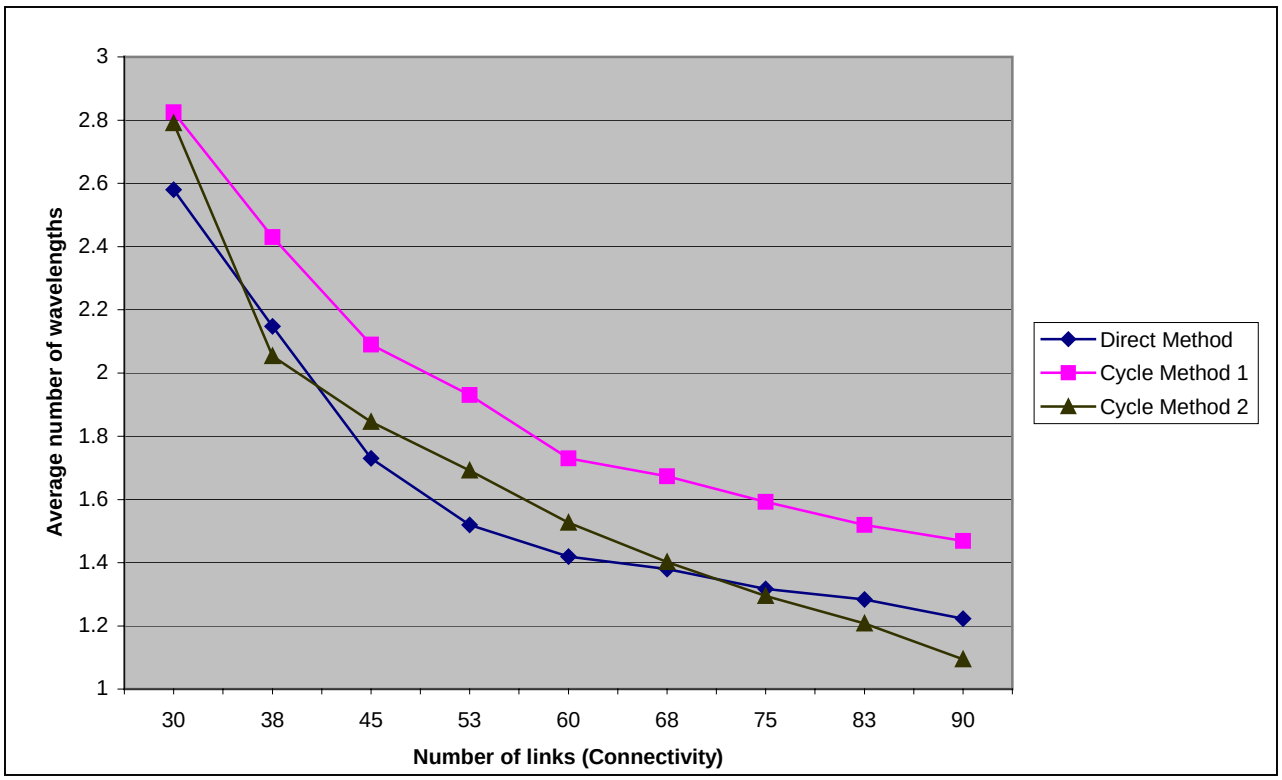


Figure 4.13: Average number of wavelengths on a link for 20-node network with variable connectivities.

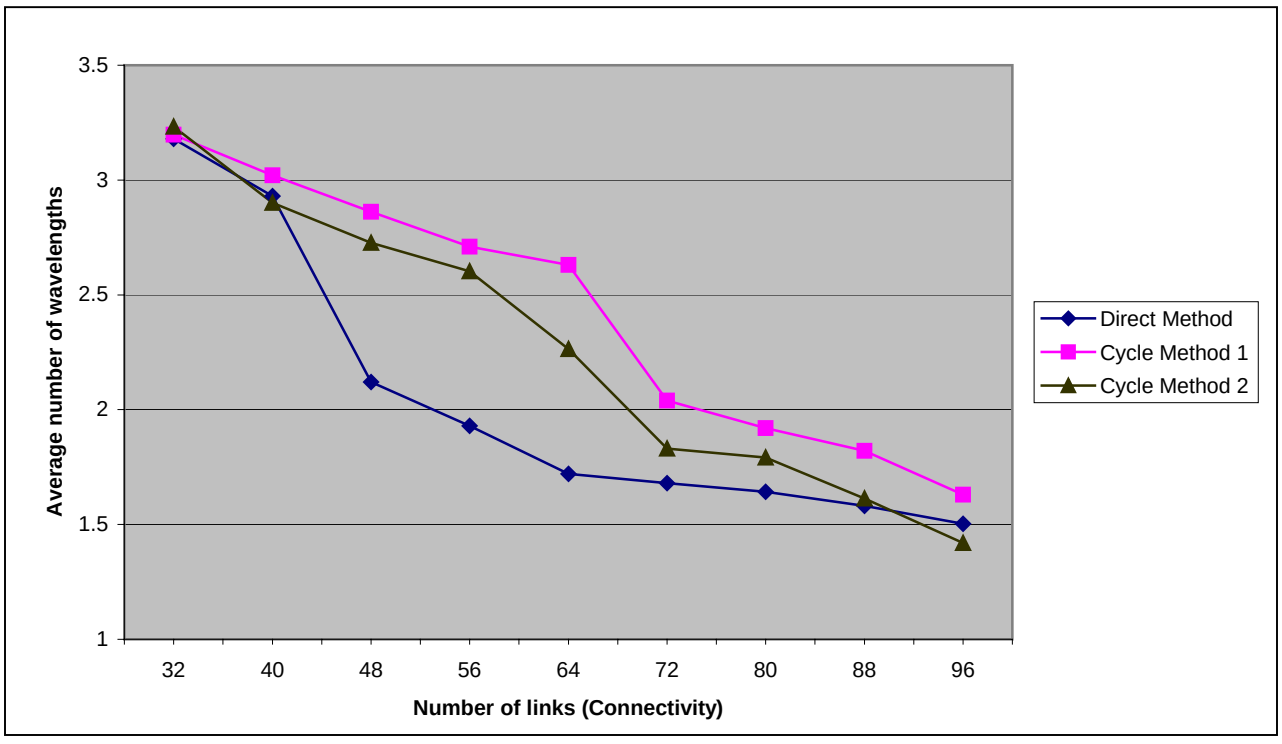


Figure 4.14: Average number of wavelengths on a link for 25-node network with variable connectivities.

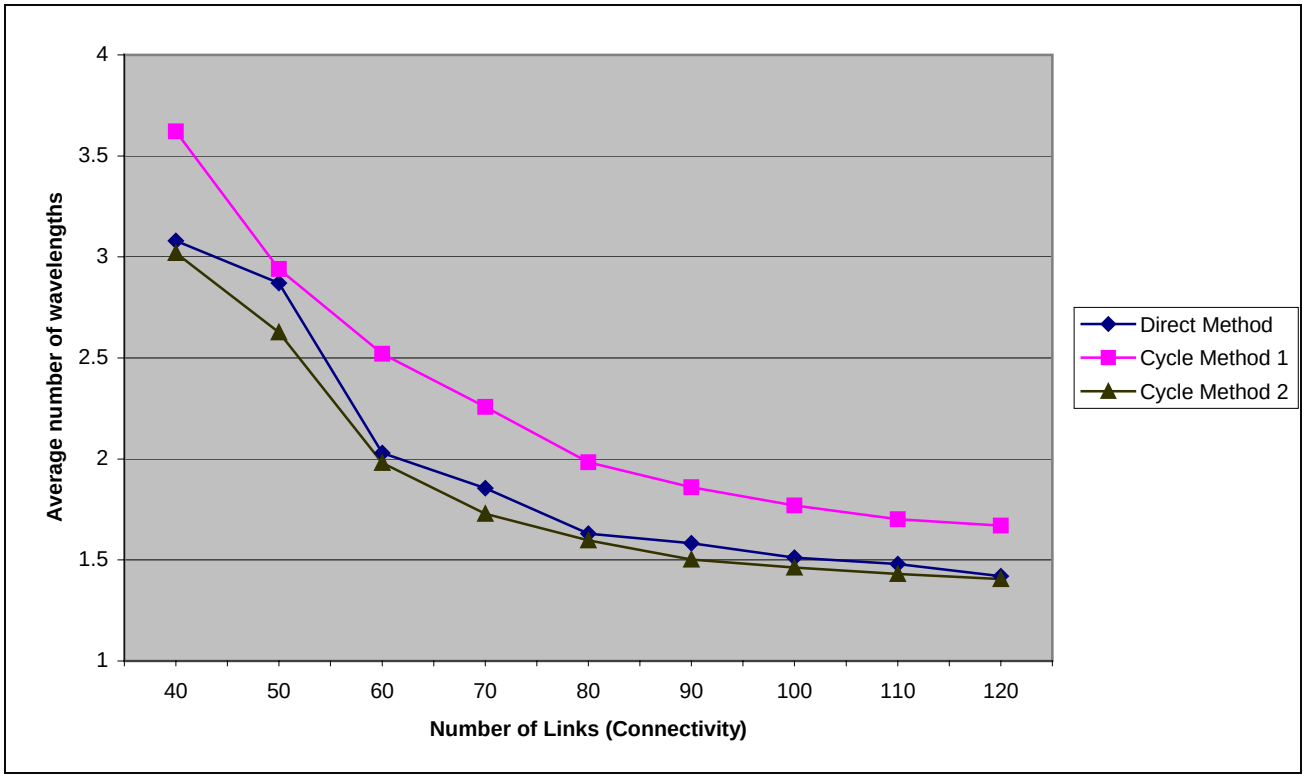


Figure 4.15: Average number of wavelengths on a link for 30-node network with variable connectivities.

4.2.4 Total Number of Link Channels

This parameter is defined as the total number of used channels (wavelength on a link) used in the original topology to implement the embedding. It is computed by adding up all the number of wavelengths needed on all the physical links of the network. Clearly, the lower the total number of link channels used, the better is the embedding. The results for this parameter for networks with different numbers of nodes are shown in the Figures 4.16 - 4.20. The figures show the total number of link channels for 10-node, 15-node, 20-node, 25-node and 30-node arbitrary networks, respectively, with variable connectivities. From the graphs, we can easily observe that smaller networks with lower connectivities are best served using the Direct Method. As the connectivity is increased, the Cycle Methods tend to perform better. This is because with larger connectivity one can find more cycles satisfying the conditions of the two Cycle Methods. For

large networks with higher connectivities, Cycle Method 2 gives the best embedding results for this parameter.

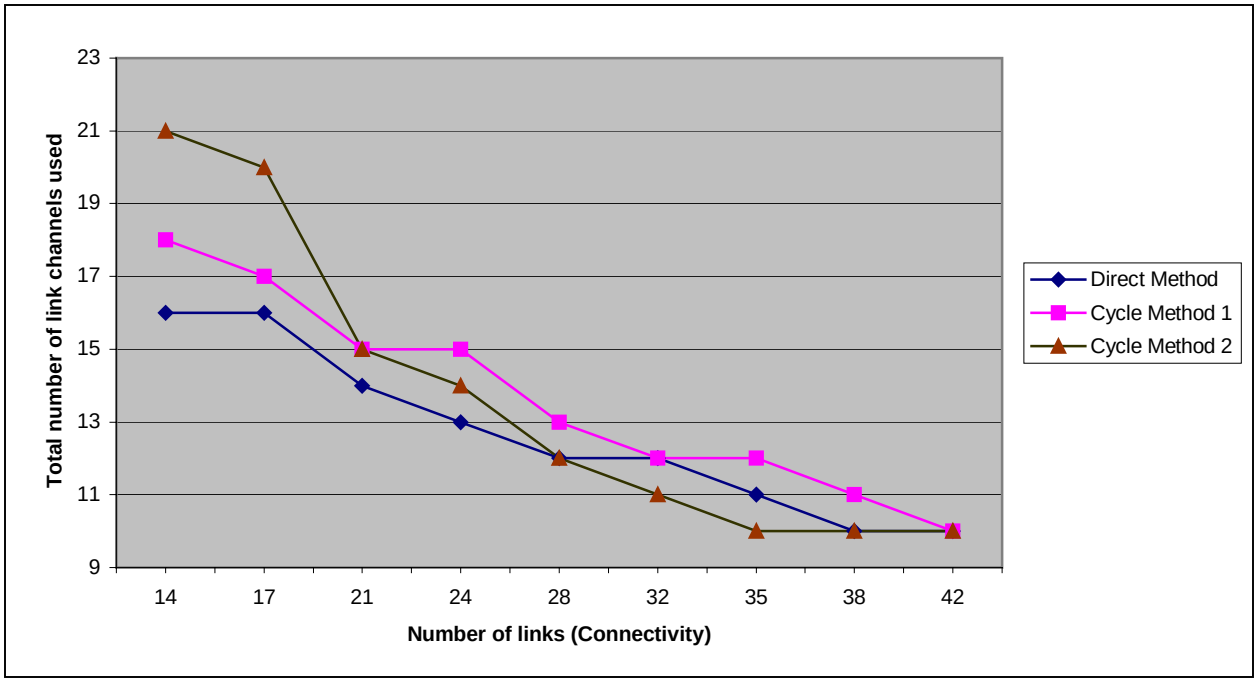


Figure 4.16: Total number of link channels used for 10-node network with variable connectivities.

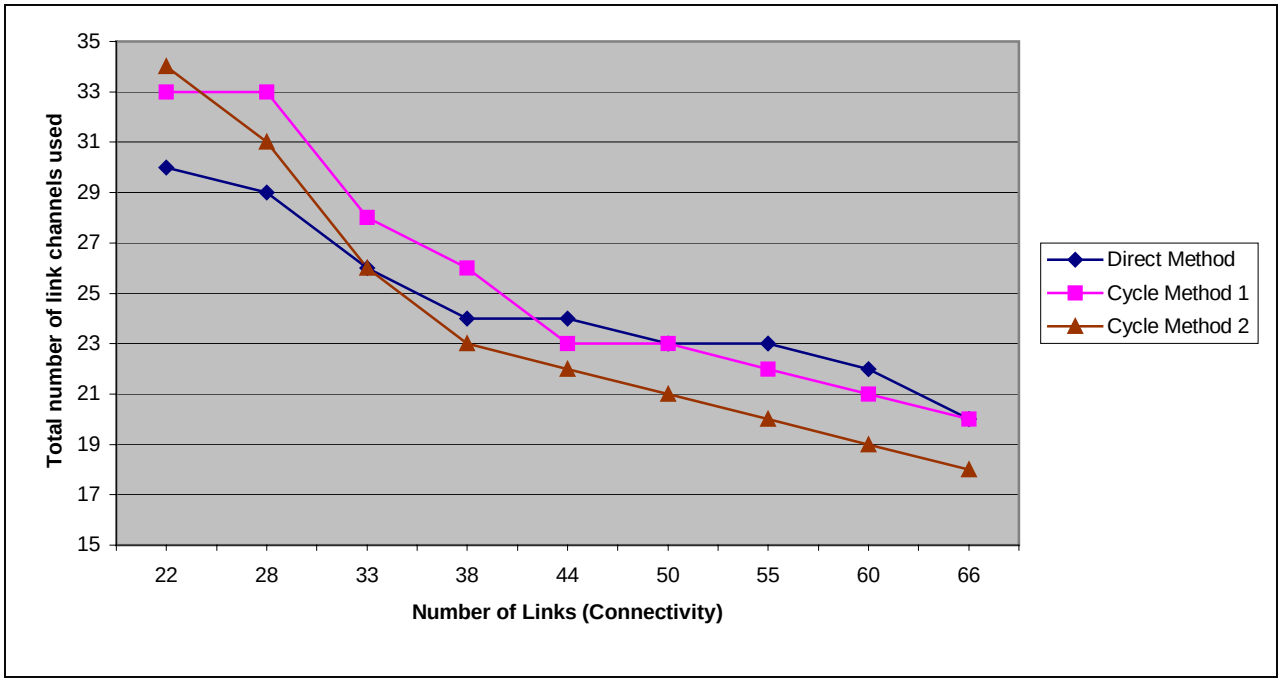


Figure 4.17: Total number of link channels used for 15-node network with variable connectivities.

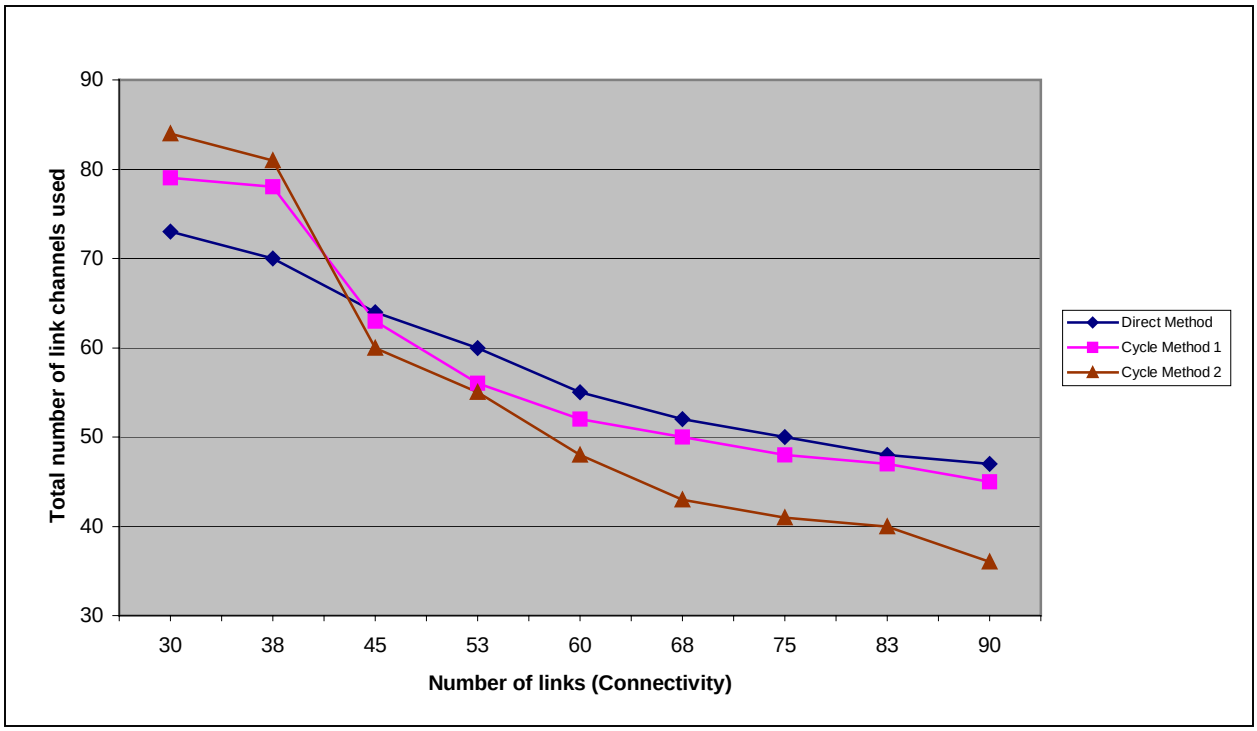


Figure 4.18: Total number of link channels used for 20-node network with variable connectivities.

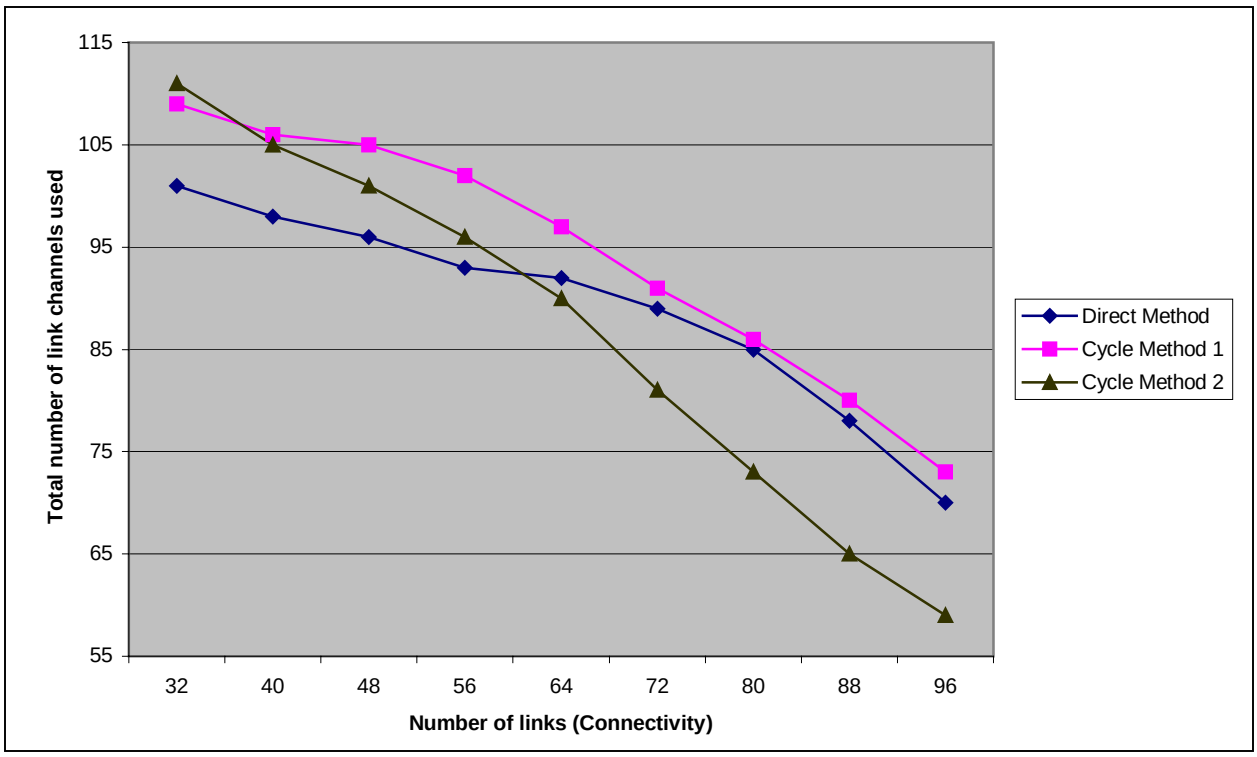


Figure 4.19: Total number of link channels used for 25-node network with variable connectivities.

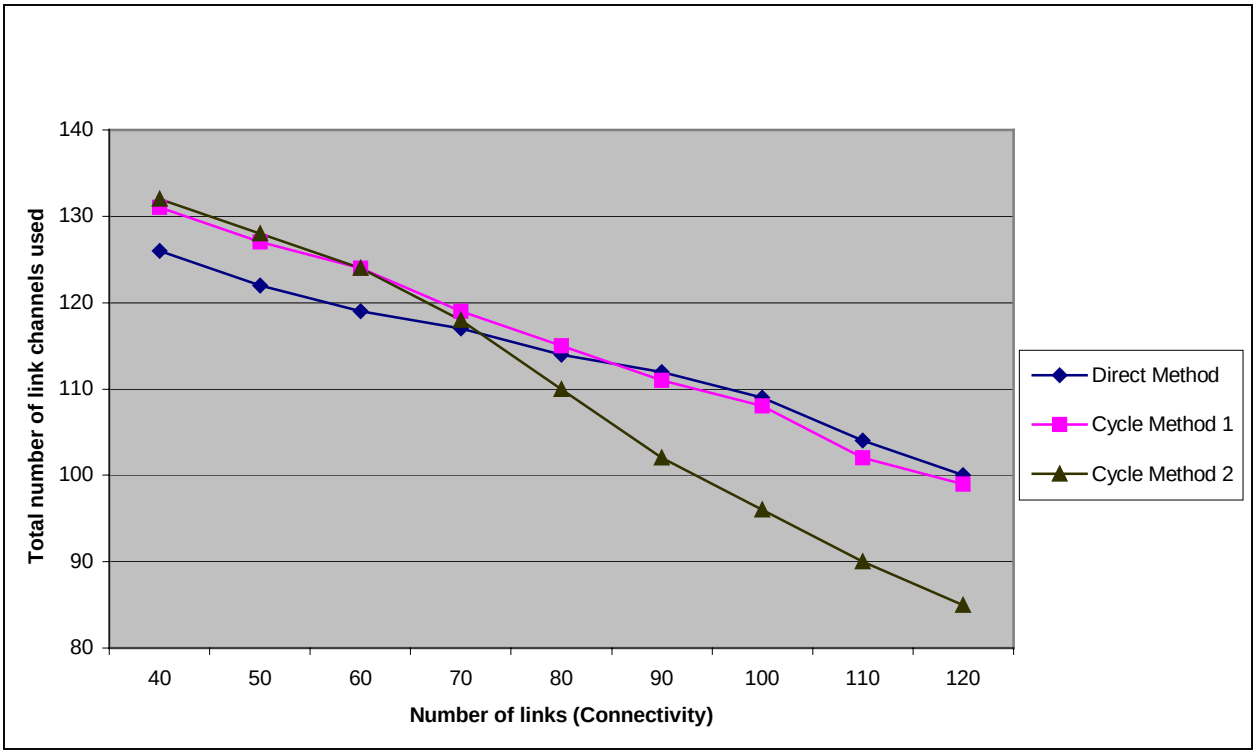


Figure 4.20: Total number of link channels used for 30-node network with variable connectivities.

4.3 Comparison between the Proposed Algorithms and Algorithms in the Literature

Considering to the literature, there is just one algorithm that was proposed for the purpose of embedding a hypercube into an irregular topology with any number of nodes. This algorithm, described in [10], uses the concept of unidirectional incomplete hypercubes. We compare our proposed algorithms with the algorithm described in [10] with respect to the parameters we used in the previous section. We ran all the algorithms on existing "real" networks like NSFNet, vBNS network and USA long haul network (refer to Appendix B, Figure B.2) and on a synthesized experimental network (refer to Appendix B, Figure B.1) and compared the performances of the algorithms. Figure 4.21 shows the comparative results for average physical path length between arbitrary virtual node pairs for the four different networks. Figure 4.22 shows the average virtual path lengths between the hypercube neighbors. Figure 4.23

compares the average number of wavelengths per link and Figure 4.24 compares the total number of wavelengths in each of the four different networks mentioned above.

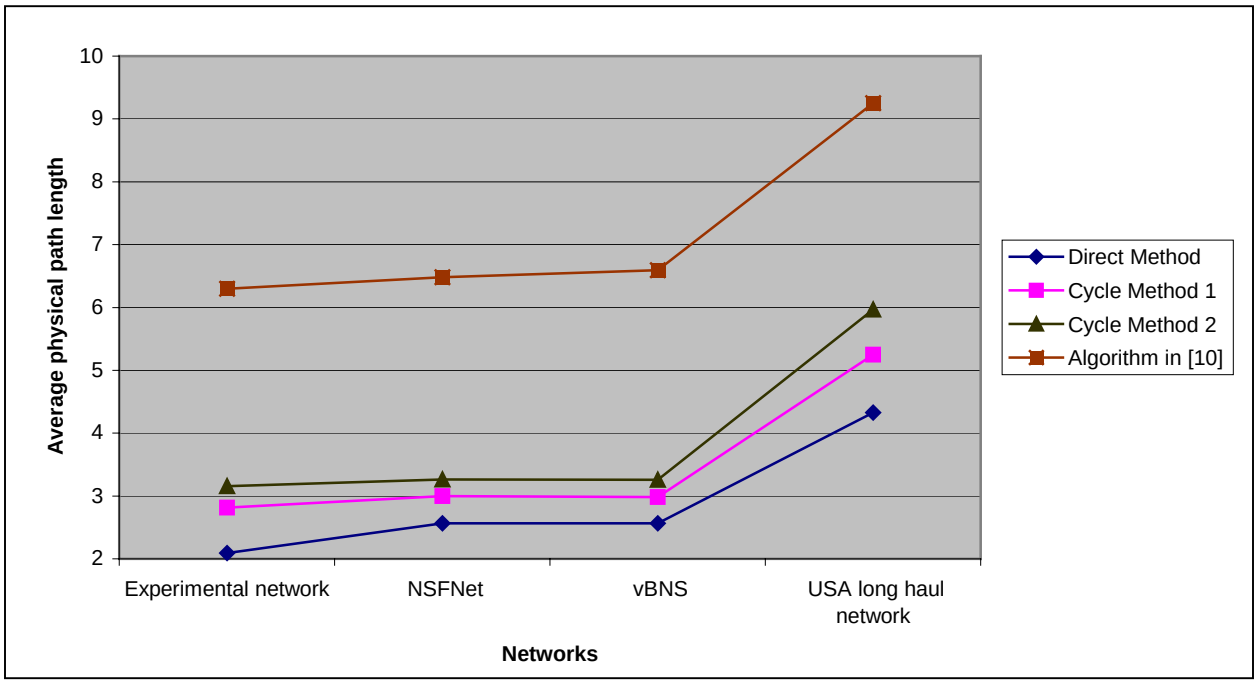


Figure 4.21: Comparison of proposed algorithms with the algorithm in [10] with respect to average virtual path lengths between any source-destination pair for different networks.

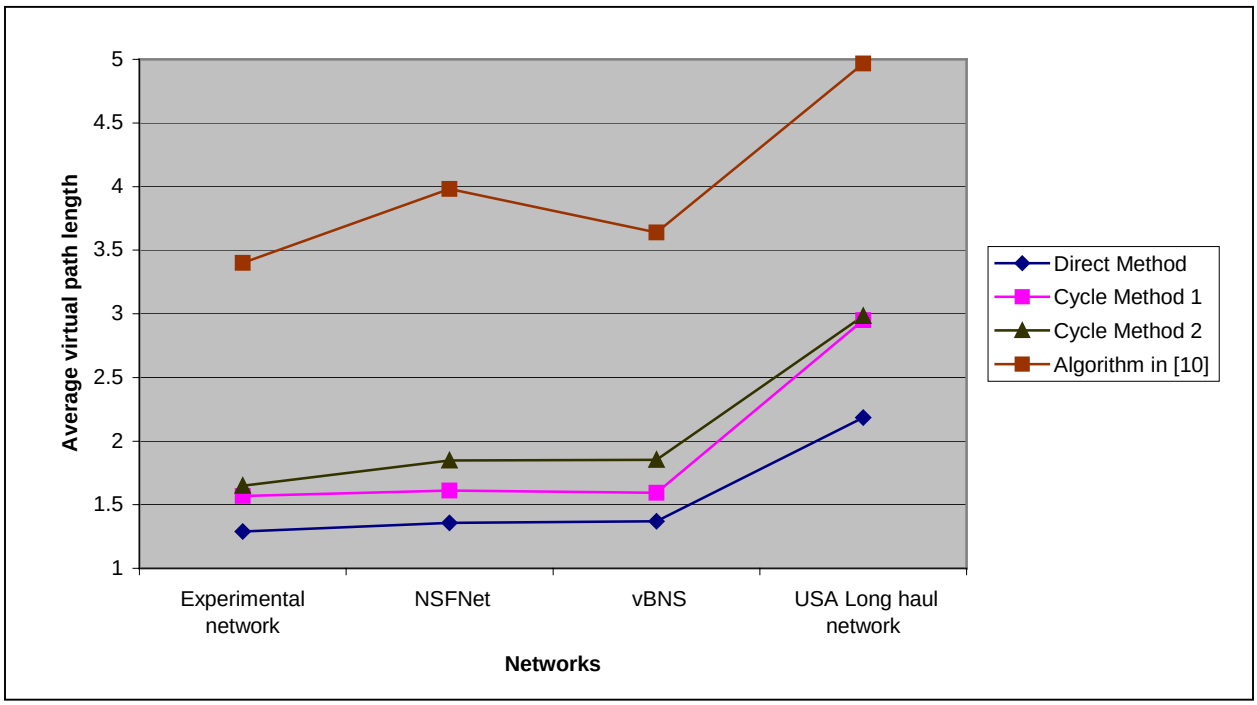


Figure 4.22: Comparison of proposed algorithms with the algorithm in [10] with respect to average physical lengths for realization of virtual links for different networks.

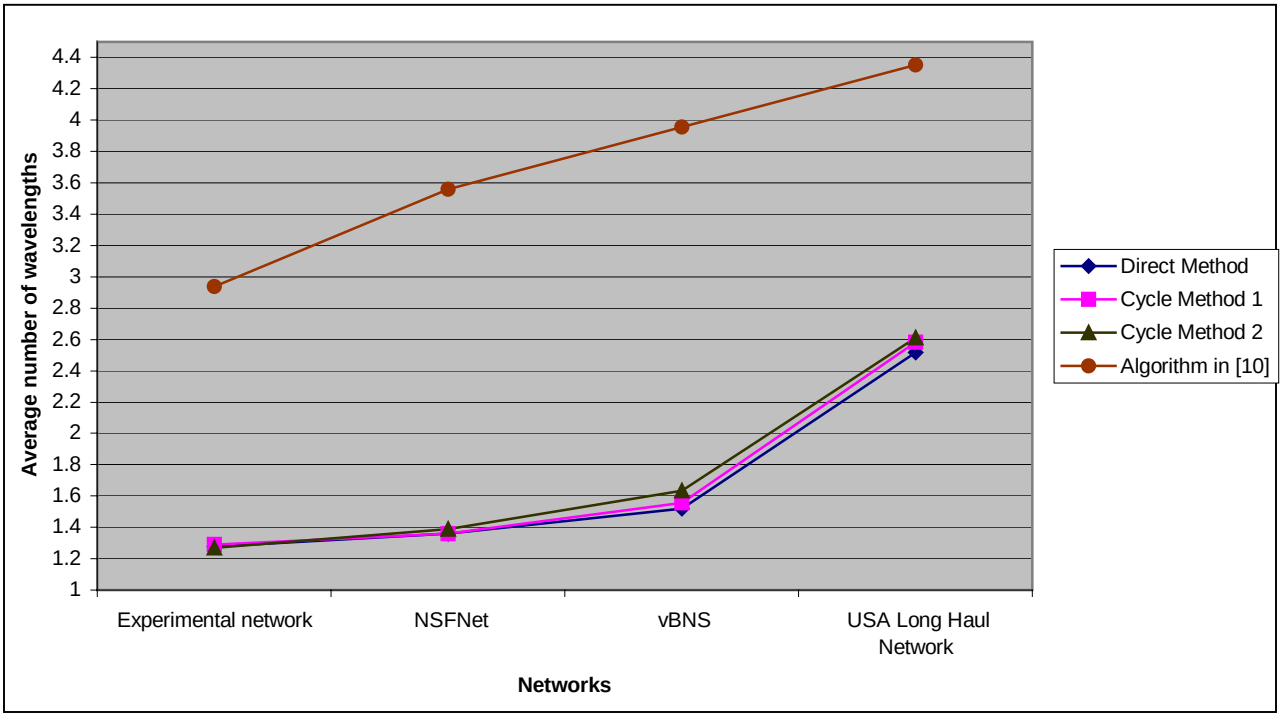


Figure 4.23: Comparison of proposed algorithms with the algorithm in [10] with respect to average number of wavelengths per link for different networks.

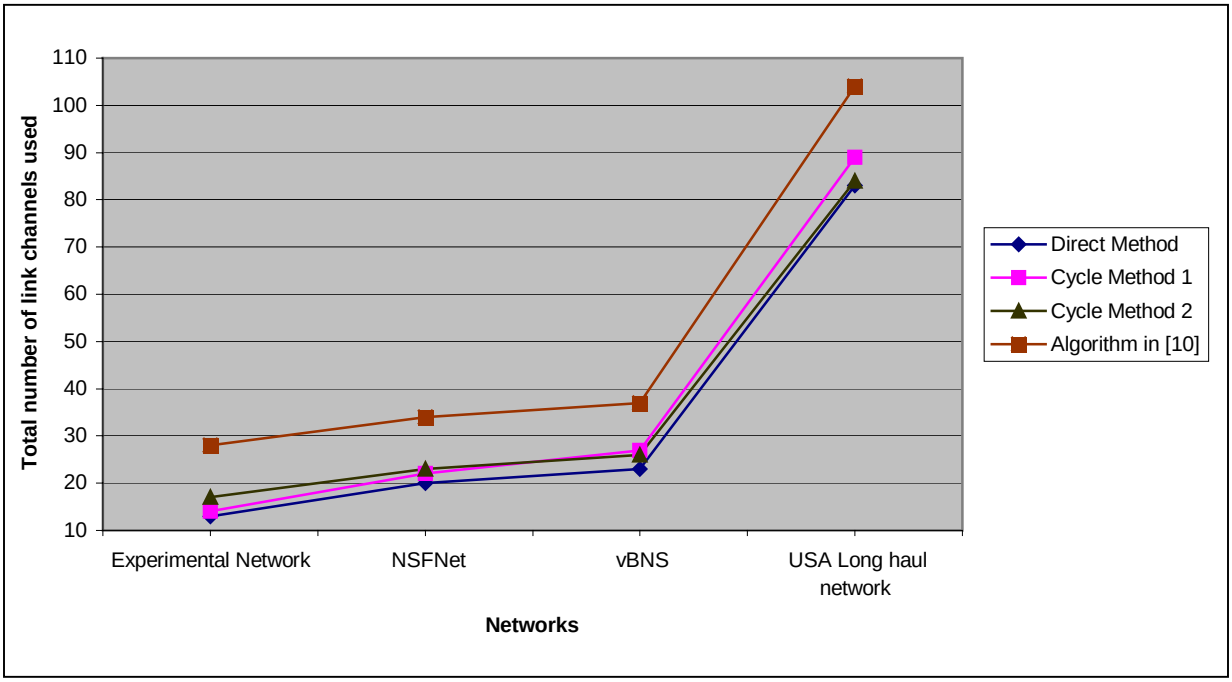


Figure 4.24: Comparison of proposed algorithms with the algorithm in [10] with respect to total number of link channels used for different networks.

From the above graphs, we can observe that the three algorithms presented in this thesis clearly outperform the algorithm described in [10] with respect to all the parameters considered. The algorithm described in [10] uses unidirectional links to connect the virtual nodes in the hypercube. The use of unidirectional links not only increases the average physical path length, but also drastically increases the number of wavelengths required. In the present scenario, considering unidirectional links in optical networks means a huge wastage of available fiber. Also building incomplete hypercubes which is the basis of the algorithm in [10] does not necessarily try to maximize number of the virtual links with dilation 1 mappings. The three algorithms presented in Chapter 3 consider bi-directional links and always attempt to maximize the number of virtual links with dilation 1 mappings. All the three algorithms give comparable results for all the four networks considered. The percentage decrease in the above-mentioned parameters is listed in Tables 4.1 - 4.4.

Table 4.1 Percentage decrease in average physical path length between arbitrary virtual node pairs when compared to the algorithm in [10]

Networks / Methods	Experimental Network % Decrease	NSFNet % Decrease	vBNS Network % Decrease	USA Long Haul Network % Decrease
Direct Method	106.6858	86.46536	88.98037	46.96633
Cycle Method 1	68.79875	75.20027	82.84314	42.74915
Cycle Method 2	68.79875	76.08185	83.12366	40.38841

Table 4.2 Percentage decrease in average physical lengths for realization of virtual links when compared to the algorithm in [10]

Networks / Methods	Experimental Network % Decrease	NSFNet % Decrease	vBNS Network % Decrease	USA Long Haul Network % Decrease
Direct Method	69.30233	88.13776	80.9466	46.08824
Cycle Method 1	61.06195	83.00248	61.451	24.79899
Cycle Method 2	59.29978	85.06901	60.84142	36.45604

Table 4.3 Percentage decrease in average number of wavelengths per link when compared to the algorithm in [10]

Networks / Methods	Experimental Network % Decrease	NSFNet % Decrease	vBNS Network % Decrease	USA Long Haul Network % Decrease
Direct Method	129.6875	161.7647	160.3289	72.79587
Cycle Method 1	127.907	161.7647	154.3059	68.38235
Cycle Method 2	131.8612	156.1151	142.4632	66.57734

Table 4.4 Percentage decrease in total number of link channels used when compared to the algorithm in [10]

Networks / Methods	Experimental Network % Decrease	NSFNet % Decrease	vBNS Network % Decrease	USA Long Haul Network % Decrease
Direct Method	115.3846	70	60.86957	49.39759
Cycle Method 1	100	54.54545	37.03704	39.32584
Cycle Method 2	64.70588	47.82609	42.30769	47.61905

Chapter 5

Conclusion and Future Scope

This thesis addressed the important issue of embedding regular topologies into WDM optical networks with arbitrary topologies. More specifically we focused on embedding binary hypercubes in irregular WDM networks. The work was motivated by the need for massive communications in the control plane for the purpose of disseminating network state information. Such communications patterns can efficiently be implemented using regular topologies such as hypercubes.

We have reported three new algorithms for embedding hypercubes into irregular WDM networks. The algorithms vary in their complexity. However, all three algorithms try to minimize network resources dedicated to the embedding. The first algorithm (Direct Method) performed well consistently but was the best choice especially when network connectivity was relatively low. The other two algorithms (Cycle Method 1 and Cycle Method 2) depend on finding a cycle in the host network and then mapping that cycle to a known Hamiltonian cycle of the hypercube. These methods also worked well particularly as connectivity increased. There were differences in the performance of the three algorithms depending on the metric used for evaluations. Nonetheless, all three algorithms outperformed the only known algorithm in the literature that considered a similar problem with arbitrary number of nodes.

Finally we observe that our methodology could be easily adopted for embedding other regular topologies into irregular WDM networks. The cycle methods will be applicable if and only if the guest graph contains a Hamiltonian cycle while the Direct Method should work for all cases.

We considered the embedding of hypercubes in arbitrary networks, which are special cases of *k*-ary *n* cubes. Further work in this area of embedding different dimensions of *k*-ary *n* cubes should be considered. Also we considered unweighted arbitrary topologies. Considering weighted irregular networks will open further avenues in finding the virtual path implementation and optimizing certain parameters to get better embeddings for practical networks. Research also should be carried out in the area of developing embedding for survivable WDM optical networks.

Bibliography

[1] Dutton, H.J.R., “Understanding Optical Communications”, Research Triangle Park: IBM Corporation; 1998.

[2] Kartalopoulos, S.V., “Introduction to DWDM Technology: Data in a Rainbow”. New York: IEEE Press; 1999.

[3] Mukherjee, B., Optical Communication Networks, McGraw-Hill, New York, NY, 1997.

[4] Ramaswami, R. and Sivarajan, K.N., “Design of Logical Topologies for Wavelength-Routed Optical Networks”, IEEE Journal on Selected Areas of Communication, pp. 840-851, 1996.

[5] Cisco Systems, “Introduction to DWDM for Metropolitan Area Networks”, http://www.cisco.com/univercd/cc/td/doc/product/mels/dwdm/index.htm, 1999.

[6] Chaffee, D.C., “The Evolution of DWDM”, www.ciena.com.

[7] Pankaj, R.K. and Gallager, R.G., “Wavelengths Requirements of All-Optical Networks”, IEEE/ACM Transactions on Networking, pp.269-280, 1995.

[8] Zang, H., Jue, J.P., Sahasrabuddhe, L., Ramamurthy, R. and Mukherjee, B., “Dynamic Lightpath Establishment in Wavelength-Routed WEM Networks”, IEEE Communications Magazine, September 2001.

[9] Williams, K.A. and Du, D.H.C, “Efficient Embedding of a Hypercube in an Irregular Network”, Technical Report TR91-016 Dept of Computer Science and Engg, Univ of Minnesota, 1991.

[10] Tan, S. T. and Du, D.H.C., “Embedded Unidirectional Incomplete Hypercubes for Optical Networks”, IEEE transactions on Communications, vol. 41, no. 9, Sept 1993.

[11] Heun, V. and Mayr, E.W., “A General Method for Efficient Embeddings of Graphs into Optimal Hypercubes”, Institute of Information Technology, Univ of Munchen, Germany.

[12] Pascu, S. and El-Amawy, A. “Non Blocking Routing and Wavelength Assignment in Ring and Torus WDM networks”, LSU, Tech Report #0507/SP, May 2002.

[13] Chlamtac, I. *et al.*, “Lightnet: Lightpath Based Solutions for Wide Bandwidth WANs”, IEEE INFOCOM, vol. III, pp. 1014-1021, 1990.

[14] Katseff, H.P., “Incomplete Hypercubes”, IEEE Transactions Computers, vol. C-37, pp. 604-608, May 1998.

[15] Bannister, J.A., “The Wavelength-Division Optical Network: Architectures, Topologies, and Protocols”, Ph.D. dissertation, Computer Science Department, University of California, Los Angeles, CA, 1990.

[16] Bijja, P. “Wavelength Assignment in All-Optical Networks for Mesh Topologies”, Master’s Thesis, Electrical and Computer Engineering Department, Louisiana State University, Baton Rouge, 2002.

[17] El-Amawy, A. and Pascu, S., “Non-Blocking WDM Optical Networks”, provisional patent application, serial number 60/382,382, Patent pending.

[18] Saad, Y. and Schultz, M.H., “Topological Properties of Hypercubes”, IEEE Transactions on Computers, vol. 37, no. 7, pp. 867-872, July 1988.

[19] Frank, H. and Chou, W., “Routing in Computer Networks”, Networks, vol. 1, pp. 99-112, 1971.

[20] Brackett, C. A., “Dense Wavelength Division Multiplexing Networks: Principles and Applications”, IEEE Journal on Selected Areas of Communications, vol. 8, no. 6, pp. 948-964, August 1990.

[21] Bannister, J.A., Fratta, L. and Gerla, M., “Topological Design of the Wavelength-Division Optical Network”, INFOCOM 1990, vol. 3, pp. 1005-1013, June 1990.

[22] Vetter, R.J., Williams, K.A. and Du, D.H.C., “Topological Design of Optically Switched WDM Networks”, Technical Report TR91-012, University of Minnesota Technical Report, 1991.

[23] Ramamurthy, B. and Mukherjee, B., “Wavelength Conversion in WDM Networking”, IEEE Journal on selected areas in Communication, vol. 16, no. 7, pp. 1061-1073, September 1998.

[24] Venugopal, K.R., Rajan, E.E. and Kumar, S.P., “Performance Analysis of Wavelength Converters in WDM Wavelength Routed Optical Networks”, Proceedings of the Fifth International Conference on High Performance Computing, 1998.

[25] Ramaswami, R. and Sivarajan, K.N., “Routing and Wavelength assignment in All-Optical Networks”, IEEE/ACM Transactions on Networking, vol. 3, pp. 489-500, October 1995.

[26] Hong, J., Mehlhorn, K. and L. Rosenberg, L.A., “Cost Trade-offs in Graph Embeddings, with Applications”, Journal of the Association for Computing Machinery, vol. 30, no. 4, pp. 709-728, October 1983.

[27] Wu, Y.A., “Embedding of Tree Networks into Hypercubes”, Journal of Parallel and Distributed Computing, vol. 2, pp. 238-249, 1985.

[28] Labourdette, J.P. and Acampora, A.S., “Wavelength Agility in Multihop Lightwave Networks”, INFOCOM 90, pp. 1022-1029, 1990.

[29] Chlamtac, I., Ganz, A. and Karmi, G. “Purely Optical Networks for Terabit Communication”, INFOCOM 89, pp. 887-896, 1989.

[30] Nissen, M., “Basic Graph Algorithms”, http://www.mpi-sb.mpg.de/~marco/diplom/node25.html, 1998.

[31] Fratta, L., Gerla, M. and Kleinrock, L., “The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design”, Networks, vol. 3, pp. 97-133, 1973.

[32] Pascu, S. and El-Amawy, A., “A Tight Bound on Number of Wavelengths for Non-Blocking, All-to-All Communications in the WDM Hypercube of Even Dimensionality”, Journal of Lighwave Technology, under review.

Appendix A

Depth First Search Algorithm

The Depth First Search (DFS) algorithm can be summarized as follows:

- Begin with a starting node.
- Visit each of the nodes adjacent to the current node calling the depth first search algorithm for each node that has not been visited.

The pseudocode is shown below:

```
DFS (G)
        {
                for each vertex u ε v (G) do
                        color [u] ← WHITE;
                for each vertex u ε v (G) do
                        if (color [u] = WHITE) then
                                DFS_VISIT (u);
        }
DFS_VISIT (u)
        {
                color [u] = GREEN;              /* White vertex u discovered */
                for each v ε Adj [u] do
                        if (color [v] = WHITE) then  /* Explore all edges (u, v) */
                                DFS_VISIT (v);
```

```
                color [u] = RED;                    /* finished u; color u RED */
        }
```

At the start, all nodes are white and at the end, all nodes are colored red.

This algorithm can be used to find cycles in arbitrary networks. A DFS tries to find paths from a starting node to any other node in the network. The DFS stops when it visits the starting node again. When the DFS visits the same node again, it is essentially a cycle. We use this property of the DFS algorithm to find cycles in the network. We select only the cycles that satisfy certain conditions as discussed in Chapter 3 for the Cycle Method 1 and Cycle Method 2 algorithms.

Appendix B

Network Topologies

We used certain network topologies in this thesis like the NSFNet, vBNS, USA Long Haul Network and an Experimental network. The NSFNet and vBNS topologies have been shown in Figure 3.1 and 3.7 respectively. The other two topologies used in our work i.e. the Experimental network and the USA Long Haul Network. These networks are shown below in Figures A.1 and A.2 respectively. These four networks were used to compare our proposed algorithms with the algorithm in [10].

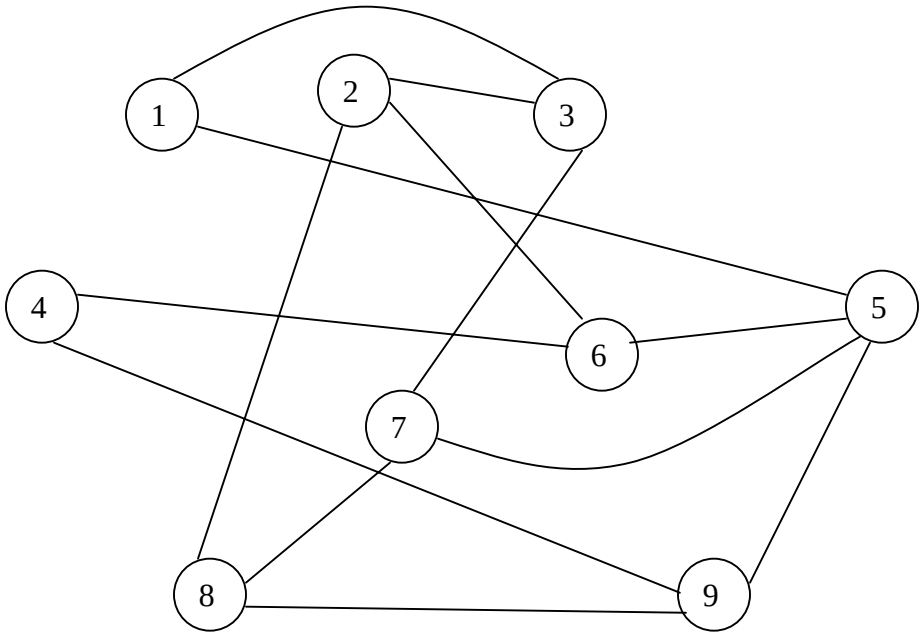


Figure B.1: 9-node synthesized Experimental Network

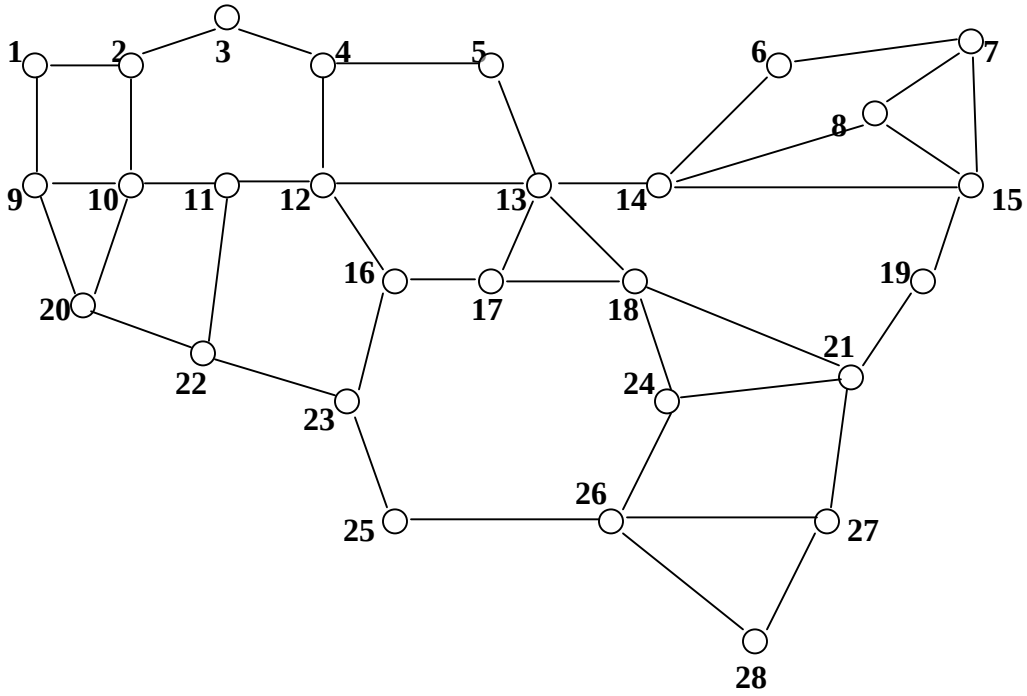


Figure B.2: 28-node USA Long Haul Network

Vita

Guru Prasad P. Kithlanagamangala was born in the city of Bangalore in the Southern Indian state of Karnataka. He graduated from High School in June 1996. In the Fall of 1996, he joined his Bachelor of Engineering in Telecommunications at Sir. M. Visvesvaraya Institute of Technology, Bangalore University, Bangalore and graduated in the year 2001 with Distinction.

He worked as a lecturer in the Department of Electronics and Communications, Sir M. Visvesvaraya Institute of Technology, Bangalore University during the first half of 2001.

He joined the Department of Electrical and Computer Engineering, Louisiana State University in the fall of 2001 to do his graduate studies and is expected to get his Master of Sciences degree in the fall of 2003.