uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Wail Mardini

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Planar Chordal Cycles: Their Construction, Applications and Analysis in Mesh Networks

TITRE DE LA THÈSE / TITLE OF THESIS

Oliver Yang

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

P. Flocchini

A. Williams

C.H. Lung

G. Takahara

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Planar Chordal Cycles: Their Construction, Applications and Analysis in Mesh Networks

by

Wail Mardini

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfilment of the requirements
for the PhD degree in Computer Science

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

September 2006

Your file Votre référence
ISBN: 978-0-494-25883-5
Our file Notre référence
ISBN: 978-0-494-25883-5

Dedication

To my family.

Holy Bible
Phi 4:6 - Be careful for nothing; but in every thing by prayer and supplication with thanksgiving let your requests be made known unto God.

الكتاب المقدس –
فيلبي4:6 - لاَ تَهْتَمُّوا بِشَيْءٍ، بَلْ فِي كُلِّ شَيْءٍ بِالصَّلاَةِ وَالدُّعَاءِ مَعَ الشُّكْرِ، لِتُعْلَمْ طِلْبَاتُكُمْ لَدَى اللهِ.

Abstract

We propose a novel idea for cycles that is based on the network topology in planar graphs. Using the graph theory terminology of chordal graphs, we call those cycles Planar Chordal Cycles (PCCs). Sufficient conditions are formulated to guarantee the existence of PCCs to cover a set of faces in the graph. These conditions to form a cycle are then mapped to the dual domain to reduce the complexity of cycle construction so that it is easier for identification and construction. This mapping is made possible by using a novel idea of constructing a dual graph that can capture the face connectivity in the original graph (and we call it the Modified Cycle Graph (MCG)). An algorithm based on some searching techniques and criteria is proposed to use those conditions to construct cycles with different objectives and properties. The application of PCCs in multicasting and mesh network protection are discussed. An efficiency and performance study has been conducted to compare this technique to other techniques.

We have also analyzed performance measures for network cycle-based protection mechanisms. We present a detailed study and analysis on the restorability of a single protection cycle covering single-domain networks, and we also present the corresponding redundancy and protection path analysis. For two-domain networks that use a single cycle in each domain, we present an analytical model that extends the single domain idea by considering the shared edges between the two domains. Limited results have been obtained in special cases for more than two domains.

Acknowledgment

I gratefully thank my supervisor, Dr. Oliver Yang, for his great help and advice during the course of my research in addition to his help in improving my writing skills.

I would like to thank the following universities and organizations for their partial financial support in completing my degree: Jordan University of Science and Technology (JUST), Irbid – Jordan, National Science and Engineering Research Council (NSERC) and industrial partners, through the Agile All-Photonic Networks (AAPN) Research Network, as well as the FGPS, the CUPE Local 2626, and the Graduate Student Association (GSAED) at the University of Ottawa .

Finally, I would like to thank my parents, brother and sisters as well as my friends and colleagues in Ottawa/Canada for their encouragement and support.

Table of Contents

List of Figures

List of Tables

Table of Acronyms and Abbreviations

		Page of 1st appearance
AC	All Cycles	55
C(G)	Cycle graph of G	19
DCSP	Degree Constrained Steiner tree Problem	4
DPRing	Dedicated Protection Ring.	7
MC(G)	Modified cycle-graph of graph G	20
MC	Multicast Cycle.	48
MCG	Modified Cycle Graph.	13
MON	Multicasting in Optical Networks	2
MST	Minimum Steiner Tree	2
OBS	Optical Burst Switching	2
OCh	Optical Channel.	5
OLS	Optical Label Switching	2
OMS	Optical Multiplex Section.	5
PCC	Planar Chordal Cycle	13
PCMCG	Planar Cycles using MCG	39
p-cycles	preconfigured protection cycles	7
PMPCC	Protected Multicast using PCC.	50
RBP	Random Backup Path.	88
SBP	Shorter Backup Path.	88
SPH	Shortest Path Heuristic	2
SPRING	Shared Protection RING.	7
TBP	Two Backup Path.	88

Table of Notations and Symbols

		Page of 1st appearance
a_{ij}	The number of edges common between face i and j	130
b	The number of shared edges between the subgraphs.	103
b_i	The number of unshared edges of face i with other faces	130
$c(v)$	$f(v)$ - deg(v, *Visited U Covered*)	36
C_{pavg}	The average of the protected capacities,	87
$C_{uavg}(x)$	The value of average unprotected working capacity conditioned on the minimum protection capacity on the protection path.	89
C_{uavg}	The average unprotected capacities,	87
C_{wavg}	Tthe average working capacities and finally	87
c_{wp}	The amount of link working capacity that can be protected by the protection cycle $_i$	87
c_{wui}	The part that has no protection because there is not enough protection capacity contributed by the cycle protection path	87
D'_{avg}	The average nodal degree in C(G).	20
D_{avg}	The average nodal degree of the network	113
$deg(n)$	The degree of node n	21
$deg(v,S)$	The number of edges joining node n with nodes in the set S	35
$e(n)$	The number of edges forming the corresponding face in G	20
E_i	The edge set for the subgraph g_i.	103
F	The ratio between the max-to-min link capacity on all the network links	55
$F_{cp}(x)$	The cumulative distribution function for $f_{cp}(x)$.	89
$f_{cp}(x)$	The density function for an amount of x link protection capacity.	89
f_{cw}	The probability density function for the working capacity on a link.	89
$f_X(x \mid k)$	The conditional density function of x conditioned on a given protection path with a length of k hops.	89
g_i	The subgraph i of a graph G.	103
h_i	The backup path length of link i.	117
H_x	The average backup path length for one protection cycle using model x for the backup of the straddling link.	117
H_{x_two}	The average backup path length for two protection cycles using model x for the backup of the straddling link.	118
M	The number of edges in the graph	16
M'	The number of edges in C(G).	20
m'	The number of edges in the MC(G) graph.	58
m_i	The number of edges in the subgraph g_i.	103
N	The number of nodes in the graph	16
N'	The number of nodes in C(G), which equals the number of faces in G.	20
n'	The number of nodes in the MC(G) graph.	58
n_i	The number of nodes in the subgraph g_i.	103
P_r	The ratio of maximum to average link working capacity	113
R_d	The ratio of the protection capacity to the working capacity of the network in order to achieve maximum restorability (100%) for a given network.	113
R_{d_best}	The best possible case for R_d.	114
R_{d_worst}	The worst possible case for R_d.	113

Chapter One
Introduction

Computer and data communication networks have evolved rapidly in the last two decades from simple network topologies like buses and rings to arbitrary mesh networks. The increasing usage of communication networks and the popularity of the Internet-based services is demanding more and more bandwidth from kbps in the early days to Gbps nowadays. Optical networking appears to be the solution to this as it is becoming an essential transport architecture to form the backbone network. However, optical networks require better technology and techniques to provide services as well as there own management and control.

1.1 Overview

Different network applications such as multimedia applications using digital audio and video have become essential in networking and the support for such huge transmission is essential. Rather than establishing a point-to-point path for multiple connection requests that basically need similar information, a single connection can be established to route from the source through different intermediate nodes to a set of destinations, forming the basis for what are called multicast trees. However, the network links have the possibility of failures that affect the data being carried. The loss is particularly huge in the context of optical networking. Therefore, it has become very important to protect these networks against failure, and to develop very fast techniques to restore the failure as soon as it occurs. It is also important to have efficient and quick methods to deal with the applications running over such networks. A cycle-based approach is the method that can provide redundant capacity in the form of a virtual ring (cycle) overlaid on a set of network links such that if any of the ring links failed to operate, the working traffic can find another path through the redundant capacity. It is interesting to see that the cycle can also be used for multicasting.

In the following sections, we present a literature review for the methods and approaches to several areas of work related to our research work: 1) the multicasting techniques, 2) the network survivability techniques, and 3) graph theory that allows us to formulate our

topology-based algorithms. It should be noted that our review naturally has a strong flavor in optical networking because it is the underlying technology for today's backbone network. There are many interesting issues related to the methods and approaches when deploying optical networking. However, many of the methods and approaches are applicable to traditional electronic networks as well which in fact are where they were originally developed and established. For example many of the multicast techniques in optical networks are based on trees which were originally proposed and used in electronic networks. It is interesting to note that graph theory still applies independent of the networking technologies.

1.2 Literature Review on Multicasting

Multicasting is defined as the ability of a communication network to accept a single message from an application and to deliver copies of the message to multiple recipients at different locations [SaMu00]. Multicasting research has a long history in various fields, and has been widely studied in IP networks [DoLe93, SaAz98]. The need for multicasting with higher bandwidth has been growing with the development of different applications such as multimedia applications including digital audio and video, and the developments in video conferencing.

Multicasting is equivalent to connecting a subset S of nodes in an undirected graph G, and many formulations are possible. The most popular one is the Minimum Steiner Tree (MST) problem that seeks a minimal cost subgraph such that there is a path between each pair of nodes in G. Finding an MST for any given graph G and node set S is NP-Complete [Karp72]. Many heuristics exists such as Shortest Path Heuristic (SPH) [TaMa80], and parallel version [MaBe93], and other algorithms [DoLe93, ZhHu93, ShLu95].

Different techniques have been proposed for Multicasting in Optical Networks (MON), which include wavelength routing [DiPo03], optical label switching (OLS) [ChEl00], and optical burst switching (OBS) [JeXi00]. Using light splitting, the time-consuming and expensive packet copying procedure in the electrical domain can be substituted by the quick and cheap light splitting in the optical domain.

A survey for multicasting techniques was given in [DiPo03]. Figure 1.1 shows a classification chart we have developed for the multicasting methods presented in this thesis.

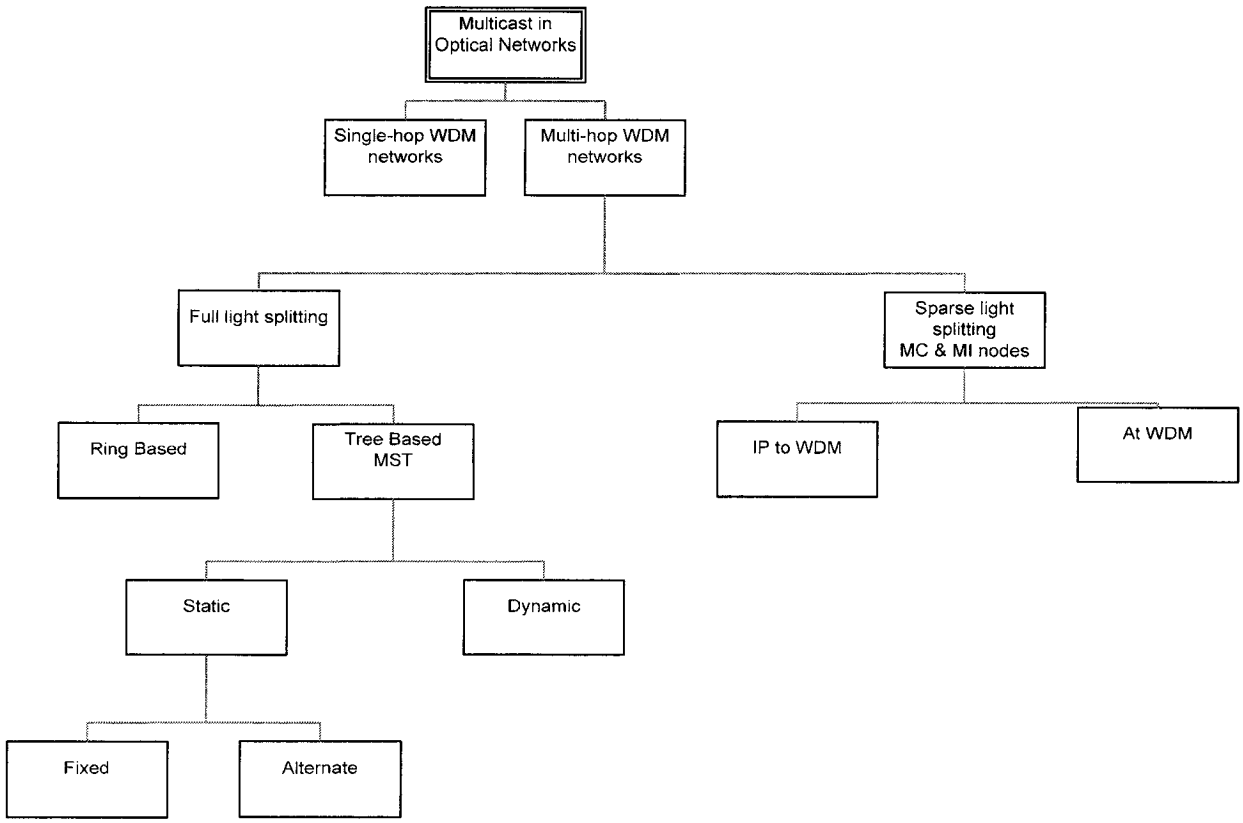


Figure 1.1: General Classifications for Multicasting in Optical Networks

According to this classification, MON can be divided into two main classes: multicast in single-hop WDM networks and multicast in multi-hop WDM networks.

1) Single-Hop Networks

In single-hop networks (which are called broadcast-and-select networks) a passive star coupler is used [LiWa01, TsKu00]. The star coupler broadcasts any information it receives from one station in a given wavelength to all the other stations in the network in the same wavelength. The stations that are members in the multicast group tune their receivers to that wavelength to receive the data. The AAPN (Agile All Photonic Networks) which has been proposed recently is an example of a single hop network [AAPN03].

2) Multi-Hop Networks

Multicast in multi-hop networks is further divided according to the light-splitting capability of the network nodes. If all the network nodes have light-splitting capability, then the broadcast is called a full light-splitting multicast [SaAz98]. Otherwise, it is called a sparse light splitting multicast [QiJe99, ZhQi99].

The full light-splitting multicast is further divided into static or dynamic for the methods using MST. Other methods will be based on the virtual rings and will be discussed later on. In the static technique, there is no consideration for the link utilization; i.e., it is independent of the number of wavelengths and the traffic load [SaAz98]. The static multicasting could be fixed or alternative. In the fixed type, the MST heuristics are used to find and prepare one MST. In the alternative type, two or more MST trees are prepared for a multicast group. Then a search algorithm is used to pick a suitable wavelength to hold the MST tree, giving rise to wavelength assignment.

In the sparse light-splitting method, the nodes are divided into Multicast Capable (MC) and Multicast Incapable (MI) nodes. The methods under this category are divided into two groups. The first group of techniques tries to map the multicast performed at the IP layer into the WDM layer. For example, the MPLS multicast [ZhQi99] [QiJe99] usually produces a forest, before a connection path is found. The GSMP protocol [DoLe93] is proposed for collaboration between IP routers and optical switches. The second group is based on implementing the multicast directly onto the WDM layer. The member-first and member-only algorithm is the algorithm in which a heuristic tree formation algorithm is used to construct a multicast tree that avoids branching at the MI switches [ZhWe00]. This will lead us to a Degree Constrained Steiner tree Problem (DCSP).

A formulation of the light-tree-based virtual topology design as an optimization problem with one of two possible objective functions has been studied [SaMu00]. For a given traffic matrix, the objective was either of the following: minimize the network-wide average packet hop distance, or minimize the total number of transceivers in the network.

Even though lots of work has been done in developing tree-based methods to implement multicasting, no multicasting method that is tree-based has discussed or presented a solution for the survivability of such a method against single failures. It is usually left to external methods to take care of the survivability.

Ring-based methods have been used extensively for network survivability. They could also be used in multicasting, thus achieving the two goals at once. Ring-based multicast using a physical ring has been studied in a few papers like [OhNo90][FeYu98] [ScSe05]. Most of the work has not emphasized on how to find the ring for a given set of nodes, but more on using

existing ring networks (e.g., WDM ring networks [ScSe05]), or on assuming a ring has already been deployed [OhNo90, FeYu98]. An algorithm [Wase91] has been proposed to construct a ring from the edges of a tree obtained from searching for the multicast nodes. However, the algorithm is not guaranteed to find the solution even though it appears to work in many cases. Furthermore, the complexity of the algorithm is very high and thus it is not scalable. On the other hand, the guaranteed method to connect a set of nodes in a cycle as discussed in the following has an exponential complexity; it is very slow even for small networks.

1.3 Literature Review on Network Survivability

The failures in the networks in general can be due to either link failures or switch (node) failures. Even though a switch failure would lead to a much higher loss of data and disruption due to the fact that the switch failure affects all links connected to it, this is unlikely to happen for many reasons. As most of the backbone network these days is optical, we will provide some of these reasons when we review below the failures in optical networks especially fiber cuts.

Even though the estimated lifetime for any mile of cable is about 228 years before its failure[ToNe94], different studies (e.g., [VePo02]) indicate that metro networks annually experience 13 cuts for every 1,000 miles of fiber, and long haul networks experience 3 cuts for 1,000 miles of fiber. Thus, the main reason for network failure is the fiber cut.

Reducing the fault recovery time is the main objective in introducing the protection mechanisms at the WDM layer targeting the SDH/SONET recovery time of 60-100 ms [Maie02]. The two alternative implementations for the recovery mechanisms at the WDM layer are implemented either at the OCh (Optical Channel) sublayer or at the OMS (Optical Multiplex Section) sublayer. In the OCh sublayer methods to protect the whole light-path are implemented (path protection), while in the OMS sublayer the methods that protect the links of the network (link protection).

According to a previous classification [Maie02], with some adaptation shown in Figure 1.2, the survivability techniques can be divided into two main classes: pre-planned protection and provisioning protection. In this thesis, we are interested in pre-planned mechanisms because they are much faster and this is a requirement for optical networks. Pre-planned

protections can be divided into two main classes based on the network layout (topology): ring and mesh. For either part, the techniques are further classified into link protection and path protection techniques. In the following, we give a brief idea of each technique, simple examples and a set of publications discussed the topic.

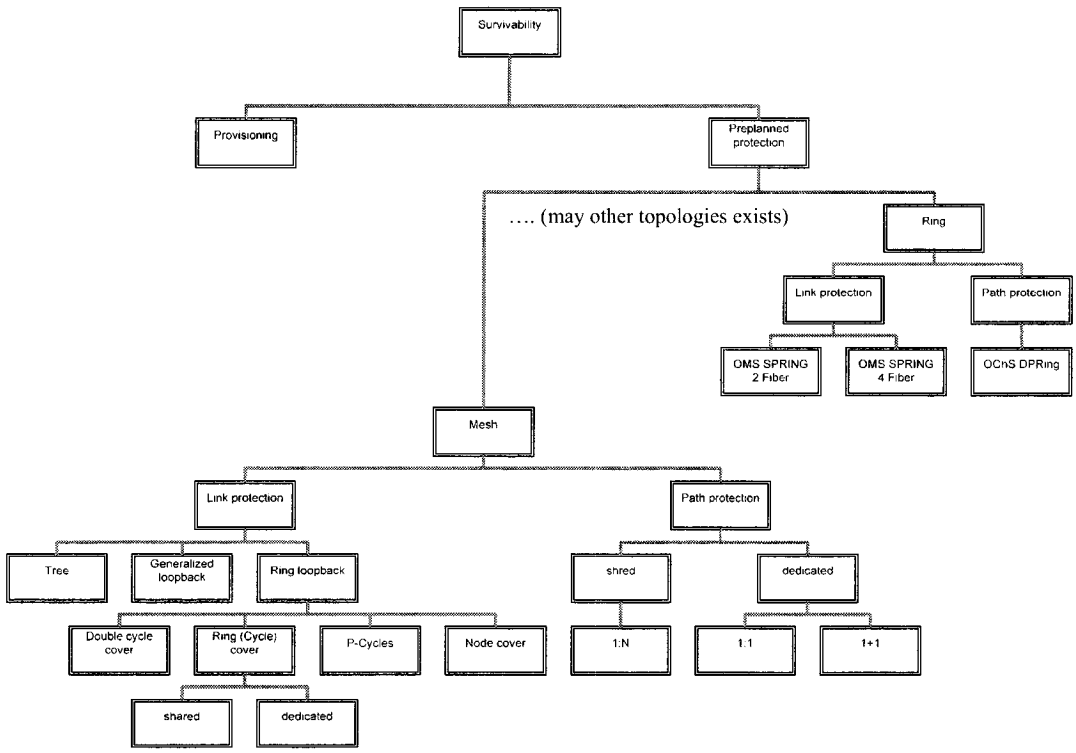


Figure 1.2: General Classification Scheme of Protection Techniques (Adapted from [Maie02])

1.3.1 Provisioning vs. Pre-planned Restorability

In the pre-planned protection, a set of routes is predetermined to protect the working traffic at the time of establishment of the working paths, while in the provisioning protection an "online methods" are used to overcome the failure as soon as it occurs. The provisioning techniques are more efficient than the pre-planned techniques in terms of capacity since we do not reserve any capacity during the normal network operation [AsSh01]. Such techniques use the available capacity to compute new routing paths in case a failure occurs[KoSu02].

In the pre-planned protection, we reserve a capacity to be used only when a failure occurs; this capacity is not used in the normal network operation. This is much faster and depends only on the failure detection time and the switching time to the predetermined paths. Since we are dealing with optical networks, the amount of traffic is usually huge and any failure

would result in a large amount of data loss. Any time spent in the computation of new routing paths for the disrupted traffic using the provisioning techniques will significantly affect the network operation [LuAn02]. Thus having a pre-planned protection technique is a very favourable and widely used option in optical networks.

The pre-planned protection techniques are classified into link-based and path-based protection techniques for both the mesh and the ring topology designed networks.

1.3.2 Link and Path Protection for Ring Networks

In the link protection techniques for the ring designed optical networks (e.g. SONET rings), the techniques used are OMS SPRING (Shared Protection RING) with two fibers and OMS SPRING with four fibers. For the path protection techniques, OChS DPRing (Dedicated Protection Ring) techniques are used [Maie02]. (See the introduction for this section 1.3 for more information about OMS and OChS)

1.3.3 Link and Path Protection for Mesh Networks

Similar to the ring networks, the protection in mesh networks can be classified into two main classes, link protection or path protection. We discuss each category and the different methods underlying them.

Link Protection

In the link protection techniques for the mesh optical networks one usually "optimizes" a set of predetermined paths for each link that has working capacity. Those sets of paths are called "loopback" since one can configure a loop back path between the endpoints of each link. These paths could be a general set of independent paths [LuMe01], or paths organized as rings (cycles) that protects a set of links at the same time [Grov98]. In the ring loopback, we have different techniques based on double and single cycle cover [GaHe94][ArGr00], p-cycles and node cover. Such rings could be organized in such a way that they use dedicated capacity or shared capacity between the rings.

To find the optimal solution for cycle based schemes, we should first find all the possible cycles in the network graph, then we find an optimized solution for a given objective and set of constraints. The number of all possible cycles grows exponentially with the size of a graph

with a general topology (e.g., a mesh network). A classification for cycle enumeration methods will be presented in a later section.

P-cycles

The main idea of a p-cycle [Grov98] is to use the cycles to protect the cycle links and the straddling links in the cycle in contrast to the regular cycle (ring) methods that uses the cycles to protect the on-cycles only. The initial step in the solution requires all simple cycles to be enumerated from the network graph which has an exponential complexity as noted above. Therefore, the optimal solution for the p-cycle problem requires an optimization based on this set of all simple cycles, thus giving rise to an NP-complete problem. In addition, finding the optimal set of p-cycles that protects all the working links can be done jointly or non-jointly with the demand routing [GrDo02, GrSc02, ScSe05]. There have been a few approaches to solve this NP-complete problem in a reasonable amount of time [StGr00]. For example, a fast algorithm called the Straddling Link Algorithm (SLA) [ZhYa02a] was proposed to a find set of p-cycles for a network. An extension for the SLA has been studied [DoHe03] where different methods of cycle expansion and growth have been presented. Their method seems to perform well for the example networks presented in that paper. However, performance evaluation for general graphs was not presented. In addition, their cycles might contain links in different areas. That is, the cycle might protect links further than near links that are protected by another cycle. As an alternative, [ZhZh03] presented a greedy algorithm based on updating the working capacity after cycles have been found.

In [GrSh03] the authors tried to extend the idea of p-cycles to path-segment protection rather than only on-cycle and straddling links protection. This implies some improvement in the spare capacity usage; however, the complexity of the problem is rather increased.

Different studies have explored the effect of limiting the hop-count for obtaining better optimal spare capacity results. Specifically, the threshold of the hop-limit has been studied for p-cycles [AdSa05] and for independent paths (we call them "all-routes" later) [HeBy95]. It has been shown through examples that p-cycles require a hop-limit threshold that is around three to four hops more than those for all-routes.

Path Protection

In the path protection techniques for the mesh optical networks one usually "optimizes" a set of shared or dedicated protection capacity for each end-to-end working path. This would use the idea of 1: N for the shared techniques where one protection path is planned to protect a number of different working paths, while 1: 1 and 1+1 is used for the dedicated methods in which one path is dedicated to protect another path [RaMu99a].

1.3.4 Multi-Failure and Multi-Domain Protection

Most of the methods that have been discussed in the previous sections do not consider the probability of having more than one failure at the same time. Some works have considered formulating some optimization models for the purpose of protecting networks from multi-failure, but the complexity of such formulations is high [ScSe05][Grov03]. It is preferable to have modified protection techniques that take care of these issues.

There has been some work on multi-domains that is related to routing in multi-domains [DiMa91]. Instead of protecting the whole network with one cycle, an interesting idea has been introduced in [HuCo01] to divide optical networks into different domains and to protect each domain independently with a Hamiltonian.

1.4 Literature Review on Related Graph Theory Work

We divide our review of graph theory literature into two categories: Cycle Enumerations and Planar Graphs.

1.4.1 Cycle Enumerations

Cycle enumeration is basically finding all possible simple cycles in a given graph. Different methods have been proposed for cycle enumeration, and they can be classified into different categories depending on the underlying approach [MaDe76]:

1) Cycle vector space: This approach depends on the fact that the set of all cycles and edge-disjoint unions of cycles is a vector space [Sysl81]. The idea is to start with a cycle basis and then compute and test (all) the other vectors to find out which are cycles or not. Different algorithms differ in trying to generate as few vectors as possible that contain all the cycles [DoKr96].
2) Search algorithms: This approach relies on finding efficient algorithms to find a super set that contains all the cycles, and then to test and extract the cycles with the desirable

properties. Examples are the backtracking algorithms [HsHo72] and the search permutations algorithm [Tier70].

3) Edge-digraph: This approach first builds the edge-digraph of a digraph, and then deletes cycles of minimal degree. The procedure is repeated until no further deletion is possible. These operations can be easily generated using matrices as done in [CaGl66,Tarj73].
4) Adjacency matrix: this method first set up an adjacency matrix A to reflect the nodes direct connectivity. It then computes A^2, A^3... A^n which represents the node connectivity for paths of lengths 2,3, ... n, and using a set of related matrices and operations to compute the cycles of each possible length from 3 up to n [MaDe76].

One of our interests is the method that uses cycle graphs obtained from the plane cycle basis of a graph, first introduced by Syslo where a new graph is produced (mapped) from the "faces" of a planar graph [Sysl81]. The resulted cycle graph is then used to find (enumerate) all the simple cycles for the original planar graph based on the fact that the set of faces form a basis for all simple cycles.

1.4.2 Planar Graphs

Since most of the backbone network is representable by planar graphs (i.e., embeddable without intersection of edges) we will consider such graphs as the basis for our work.

A chord is defined in plane geometry as a "line segment joining two points on a curve" [Weis06], and a cycle chord is a line segment joining two points in a cycle but not part of the cycle. Then, a graph is said to be chordal graph if every cycle with length more than three always contains a chord.

Chordal graphs and the related idea of chordal rings have appeared in the literature and been studied for various objectives; for examples, in efficient routing [AtLe89] [KrLu95] or in wireless for network decontaminating [FlHu06]. In all such works, the network is assumed to be chordal. However, there appears to have no work done in finding chordal subgraphs out of a non-chordal graph.

A Hamiltonian cycle is a cycle which contains every node exactly once. Finding Hamiltonian cycles in planar graphs has been studied for specific classes of graphs. However, the problem is NP-Complete for a general graph, even for a 3-connected planar

graph [GaJo76]. Although the problem for a 4-connected graph has been shown to be linear-time solvable [ChNi89], the same problem for 2-connected graph still remains to be a hard problem. 2-connected graphs is the class of the graphs we will be dealing with in this thesis and will be discussed in Chapter Two and it simply mean that every two directly connected nodes have another simple path connects them.

1.5 Motivations

As mentioned, the advent of optical networking has provided enormous bandwidth to extend/enrich existing applications, and multicasting is one application that can benefit from the enhanced performance. Multicast using trees in multi-hop wide-area networks (usually called mesh networks) is a very common approach as discussed but it needs an external protection mechanism to survive against single link failures. Since a cycle is a closed loop, using cycles for multicast implies a built in protection mechanism. However, finding multicast-cycles for a given subset of nodes in a network graph is an NP-complete [BoMu82] problem as the simple/brute-force approach of enumerating all possible cycle in the network has an exponential complexity with respect to the network size. Unlike tree-based multicast that can add a node easily, it is a challenge to use the cycle approach. Thus, we are motivated to find good algorithms with polynomial complexity, if possible, for multicasting using cycles.

A failure in the optical layer will have the most impact on the services (IP, ATM etc) built on the upper layers because the time to detect a real physical layer failure from the upper layer can be quite long if one has to rely on the timeouts from all intermediate layers. Thus, having a protection scheme implemented at the lowest layer (the physical/WDM layer) would provide the best and fastest protection. Current cycle-based protection approaches have a scalability problem as reviewed when required to cover every link that the cycle surrounds. They are also not amenable to multiple failures. Therefore, we are motivated once more to improve these types of shortcomings.

As evident from the above two motivations, scalability will be a big issue in designing an algorithm that can work well with any network size. One would not (and usually cannot) enumerate all the candidates (all network paths, simple or closed) to find the optimal solution, as the number simply grows exponentially with the network size. Therefore, we would like to introduce a new approach for providing a good solution subset that most likely

contains a good sub-optimal solution, and at the same time provides a reasonably good reliability for the network. Ideally, such an approach would also allow a new optimal solution set to be re-established after any network changes.

Embedding logical topologies over a physical topology is a common networking technique to support various network protocols and functions. Multicasting and network protection as discussed above are such examples. Various cycle-based approaches such as rings, p-cycles and Hamiltonian cycles have been reviewed, but it would be good to understand their capabilities such as capacity, simplicity, propagation delay (worst for Hamiltonian cycle) and others discussed above. Also, since there have been very few papers on a comprehensive performance study for the cycle-based approaches for network protection n, we are particularly interested in providing a more comprehensive performance evaluation as well as theoretical analyses that provide closed-form formula for different network performance aspects. A case is the use of Hamiltonian cycles as a protection approach in both single and double domains, which we become interested in when we want to seek out the related protection performance of p-cycles. For example, finding a Hamiltonian cycle for a 2-connected graph is a related but challenging problem by itself as reviewed before. Thus dividing the network into domains is not only interesting but should be easier to find a Hamiltonian cycle for a smaller subgraph (a domain).

As one last point on the theoretical aspect, we are intrigued with the approach of using the dual graph domain to represent cycles, and our initial investigation has suggested that this is a viable approach, and hence we are motivated to develop a very fast algorithm based on this concept in order to meet the challenges discussed above.

1.6 Objectives

In general, we are interested in investigating the theory, analysis and application of cycle-based algorithms. Specifically, we are interested in

1) Using graph-theory based techniques and methodology to develop cycle-based methods for multicasting and protection in mesh networks.
2) Comparing different cycle-based approaches and studying their efficiency and trade-offs with respect to different network parameters.
3) Establishing a basis for theoretical analysis for performance evaluation of the cycle-based network protection techniques.

1.7 Methodology and Approaches

As suggested in the motivation discussion, we are interested in finding cycle-based algorithms to accomplish our research objectives. Our initial study has suggested that graph theoretical approaches especially the dual cycle graph concept would allow us to build a special type of cycle. We then borrow the idea of chordal planar graphs to build what we call PCC (Planar Chordal Cycles), which has similar properties to p-cycles. We start by studying how mapping between the PCC and its dual captures the properties/characteristics of this type of cycle. For example, our algorithm fully characterizes the face adjacency in term of nodes and edges. This allows us to provide a prototype algorithm based on the simple searching techniques of the Breadth-First or Depth-First search methods.

In addition to the normal assessment of the algorithm in terms of complexity, generation efficiency and accuracy, we have built into the algorithm (call the MCG (Modified Cycle Graph) algorithm) different combinations of search techniques and search criteria (e.g., cycle size limits) so that it would carry out two major applications: the multicasting and network protection, which are described as follows.

As suggested in our motivation, our multicasting will be cycle-based. We use our PCC to establish multicast cycles. We also improve its performance such as removing the extra parts of the cycle that is not required in the multicast and relaxing some original conditions of PCC to allow shorter paths along the boundary. We evaluate the performance in terms of the length of the cycle (in term of number of hops).

We then investigate the capability of PCC for network protection, one of the original motivations. Different searching criteria (such as various forms of size limits) are used to improve the performance in terms of capacity requirement for network protection. In doing so, we also need to use optimization on generated PCC candidate set. We also compare its performance with the most optimal solution (obtained by providing the set of all possible cycle as an input to the optimization model) as well as using all the possible paths in the network. As we have mentioned in the literature review, the optimal set of p-cycles that protects all the working links can be done jointly or non-jointly with the demand routing. We consider in this thesis the non-jointly model for the following reasons: working capacities are usually optimized for some objectives like load balancing and path length. In some cases,

specific schemes for routing traffic would be desired by the network management to use specific paths rather than other. Thus, we assume a given distribution for the working traffic among the network.

We also use the evaluation results of PCCs to support our theoretical study of Hamiltonian cycles which is a special case of PCCs. We shall provide more theoretical analysis in closed-form formula for different network performance aspects as well as a framework for the study of multi-domain protection. Different performance measures such as network redundancy, restorability, and average protection path length, will be defined and evaluated in terms of various parameters such as the cycle size and the max-to-min capacity ratio. These enable us to conduct other trade-off studies such as computation speed versus coverage or spare capacities.

By taking multi-domain management issue into account, where the privacy of each domain is maintained, we hope to provide pre-planned protection solutions based on the resources available in that domain. We shall first start with single domain that has generated much research interest, and then extend it to double domains. We also provide some guidance for dual failure protection.

There are several tools/techniques that we rely on to carry out our research. We have used C/C++ language [DeDe98] to program/implement our ideas and algorithms to evaluate the performance, or to provide verification to our theoretical analysis. We also use optimization packages of AMPL [AMPL06] and CPLEX [CPLE06] to allow us to determine the optimal subset of the solution set generated.

We test our algorithms on a large number of randomly generated graphs in order to provide the average performance within a give confidence interval. These are generated using LEDA libraries [LEDA06] in C/C++ language. The large number of randomly generated graphs will enable us to cover more possible future topologies and make our analysis more reliable. We also provide 95% confidence intervals to cross check the accuracy of our evaluation.

1.8 Contributions

The following are the contributions of this research work:

1) The introduction of the concept of PCC (Planar Chordal Cycles) and its mapping into dual graph domain.
2) The MCG (Modified Cycle Graph) that fully characterizes the face connectivity for a planar graph.

3) Graph-theory based algorithms that allows the fast generation of PCC.
4) The application of PCC on network protection.
5) The application of PCC on multicasting.
6) Theoretical study of single-domain and dual-domain protection using Hamiltonian Cycles
7) A comprehensive comparison among PPC, rings, p-cycles and all-routes as protection mechanism for optical mesh networks in term of restorability.

1.9 Thesis Organization

The remainder of the thesis is organized as follows. Chapter Two discusses the network model, operations and assumptions to be used throughout the thesis. Chapter Three is devoted to characterizing and generating planar chordal cycles. Chapter Four presents the use of PCCs for protected multicasting and also contains further improvements, comparisons and a performance study. Chapter Five studies the use of PCCs for network protection with a comprehensive comparison with p-cycles which in turn are compared to rings and the all-routes methods. Chapter Six presents an analytical study of cycle-based network protection. Chapter Seven concludes our findings and suggests future work.

1.10 Publications

1) Wail Mardini and Oliver Yang. "Restorability Analysis of Two Protection Cycles in Random Mesh Networks", *Proceed. IEEE Comsoc/CreateNet GOSP 2006,* San Jose, California, USA, October 1-5, 2006. GOSP session 1 October 2nd, 2006.
2) Wail Mardini and Oliver Yang. "The Hop-Count Effect on the Optimal Spare Capacity for P-Cycles, Rings and All-Routes", *Proceed. IEEE CCECE 2006.* Ottawa, Ontario, Canada, May 7-10, 2006. Session P2-5.
3) Wail Mardini and Oliver Yang. "Optimal Backup Capacity Improvements in Optical Mesh Networks", *Proceed. IEEE CISS* 2006. Princeton, New Jersey, USA, March 21-24, 2006. pp.1696-1697. Session FP-06.
4) Wail Mardini, Oliver Yang and Yihua Zhai. "Using MCG to Find PCCs in Planar Graphs", *Proceed. OFC/NFOEC* 2005. Anaheim, California, USA, March 6-11, 2005. Session JWA52.
5) Shahram Shah-Heydari, Wail Mardini and Oliver Yang. "Theoretical Analysis of Restorability of Hamiltonian Protection Cycles in Random Mesh Networks", *Proceed. IEEE ISCC* 2004, Alexandria, Egypt, June 28 – July 1, 2004. Vol.2. pp. 921 - 926.
6) Wail Mardini, Oliver Yang and G.Q. Wang, "Mapping P-cycles From Planar Graph to Cycle Graphs", *Proceed. IEEE CISS* 2004. Princeton, New Jersey, USA, March 17-19 2004. pp. 807-812.

Chapter Two
Network Models, Operations and Assumptions

In this chapter, we will provide the operations and assumptions that are common for the whole thesis. Operations and assumptions specific to an application will be discussed in the relevant chapters later on.

2.1 Graph Network Model and Definitions

For easy description and analysis of the physical network, we consider a network graph with N nodes and M edges representing the network switching devices and the cables, respectively. Each physical link (a cable that can contain multiple fibers) between two devices is represented by an edge along with some parameters between the corresponding graph nodes. The set of nodes is denoted by $V= \{v_1, v_2 ... v_N\}$ and the set of edges by $E= \{e_1, e_2, e_M\}$. Each edge can also be specified by its endpoint node labels as in edge (i, j), $e_{(i, j)}$ or simply e_{ij}. We will use edge and link synonymously as well as node and vertex. Each node i will have two parameters: D_i which is the number of edges incident at node i (degree), and C_i which is the installation cost of node i. Then we can define D_{avg} as the average nodal degree (average of all D_i's in the graph). Each link i will have the following parameters:

w_i: the number of fibers in the corresponding cable.

c_i: the installation cost of that cable.

d_i: the actual length of the cable in kilometers.

c_{wi}: the working capacity of link e_i.

c_{wpi}: the portion of c_{wi} that is protected using a given protection technique.

c_{wui}: the remaining portion of c_{wi} that is NOT protected using the given protection technique. i.e. $c_{wui} + c_{wpi} = c_{wi}$

c_{si}: the capacity reserved for protection purpose on link i.

c_{wxi}: the maximum capacity on link i. This implies $c_{wi} + c_{si} \leq c_{wxi}$

Based on the above link parameters, we shall define the following:

c_{pmax}: the maximum protection capacity on a link in the network

c_{pmin}: the minimum protection capacity on a link in the network

C_{uavg}: the average unprotected working capacity on all links in the network.

C_{wavg}: the average working capacity on a link in the network.

c_{wmax}: the maximum working capacity on a link in the network

c_{wmin}: the minimum working capacity on a link in the network

F: the ratio of maximum working capacity to minimum working capacity.

$f_{cp}(x)$: the probability density function for link protection capacities

$f_{cw}(x)$: the probability density function for link working capacities

A *simple path* is a set of successive edges and nodes with no intersection. A *simple cycle* is a simple path that is closed (i.e. where the two ends of the path meet). An *on-cycle link* is a link that lies on the cycle. A *cycle chord* (alternatively called a *straddling link)* is a link that joins two nodes on the cycle but is not an on-cycle link. A graph is called 2-connected when there is at least another path between any directly connected nodes. A graph is said to be planar when there exists a planar embedding (drawing) without any intersection of its edges. It is non-planar otherwise[1]. Note that there could be more than one planar embedding for the same graph.

A *face* is a region encircled by the edges of a simple cycle (without chords), and usually identified by the edges either clockwise or counter-clockwise in its cycle sequence starting from a specific edge[2]. It is possible that the graph under consideration is a subgraph, in which case the faces in the graph can be classified into internal and external with respect to the subgraph, e.g., both cycles 9-0-1-5-9 and 5-1-2-3-5 in Figure 2.1a are external faces of cycle 6-7-9-5-3-4-6 whereas the faces 5-3-4-5 and 6-7-9-5-4-6 are the internal faces. For simplicity, the faces can also be labeled as shown in Figure 2.1b. Face number *i* will be denoted as Face*i* in the text, or C*i* as a short-hand notation.

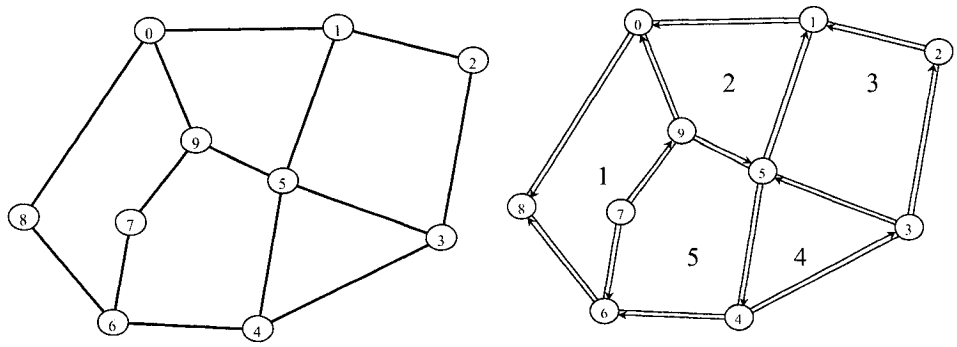
In a planar graph, the nodes (or edges) that lie on the boundary of an external face are

---

[1] The smallest classes of non-planar graphs are K_5 (complete graph with 5 nodes) and $K_{3,3}$ (complete bipartite graph with 3 nodes in each set). Kuratowski in 1930 [**BoMu82**] shows that a graph is planar if and only if has no subgraph homeomorphic to K_5 or $K_{3,3}$.

[2] In some graph theory litriture the unbounded region outside the boundary of the graph is called the *external face* and is dientifed by the boundary of the graph.

called the *external nodes* (or *edges*). Otherwise, the nodes and edges are internal. Two faces are adjacent if both faces have at least one common edge between them, and they are called *edge-jointed faces* (or *edge-disjoint* as used in graph theory although this is a somewhat confusing terminology!). If two faces have exactly one node in common (which is not part of a possible shared edge), they are called *node-jointed* (or *node-disjoint*). The set of all individual internal faces of a planar embedding of a graph is called a *planar cycle basis*. A *cycle of faces* is a set of faces where there is an ordering of the faces in which each face is adjacent to the next one and the last face adjacent to the first one. In planar graphs, $M \leq 3N-6$ [HeFu99] is a well-known relationship obtained from the fact that the smallest face in a *simple* (no multiple edges or self-loop edges) planar graph is bounded by three edges.



(a) (b)

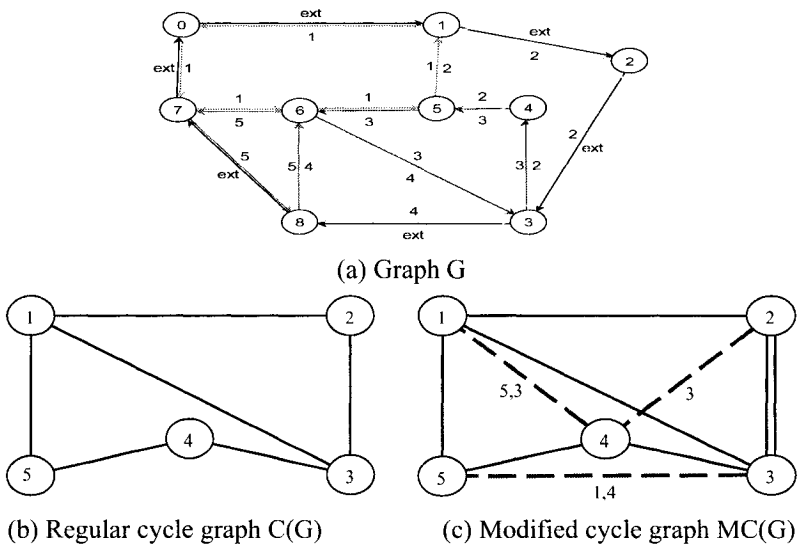
Figure 2.1: Labelling Faces in a Graph

To illustrate the above definitions and terminologies we give the example in Figure 2.1. We show in Figure 2.1b that a bidirectional link in Figure 2.1a can be represented by two unidirectional links. So for illustration purposes, where required (e.g. in later chapters), we will use these two representations interchangeably (as done in Figure 2.1a and Figure 2.1b). This figure shows an example of a planar graph with a set of nodes $V=\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and a set of edges $E=\{(0, 1), (0, 9), (0, 8), (1, 2), (1, 5), \ldots\}$. The area bounded by the edges (1, 2), (2, 3), (3, 5), (5, 1) is an internal face and denoted by Face*3* in Figure 2.1b. The nodes 5, 7, 9 are *internal nodes* while the nodes 1, 2, 3, 4, 6, 8 are *external nodes*. Likewise, the edges (1, 2) and (3, 4) are two examples of *external edges*, and the edges (1, 5) and (7, 9) are two examples of *internal edges*. Face*1* and Face*2* are adjacent (edge-jointed) since they have a common edge (0, 9). Face*3* and Face*5* are only node-jointed at node 5, and Faces Face*2* and Face*4* are also node jointed at the same node 5. Face*1* is a simple cycle, but the

simple cycle {0, 1, 2, 3, 5, 9, 0} is not a single face because it is a combination of a set of faces (Face*2* and Face*3*). Finally, faces Face*1*, Face*2*, and Face*5* form a cycle of faces.

2.2 Dual Domain Model

The previous section summarizes the terminologies in a representation which we call the "regular" graph domain. There is a dual domain representation called the cycle graph, which is a powerful representation we shall rely on in this dissertation. We first introduce the definitions and terminologies in this dual domain representation. We then present its modified version and the reasons for this modification.



(a) Graph G

(b) Regular cycle graph C(G)

(c) Modified cycle graph MC(G)

Figure 2.2: Comparison between the Cycle Graph and the Modified Cycle Graph for a given Network Example

2.2.1 Cycle Graphs and Definitions

The cycle graph[3] C(G) [Sysl81] of a planar graph G is formed by taking the faces of G to be the nodes in the new dual graph C(G). The edges of C(G) indicate the presence of edge adjacency between the faces in G. For example, v1 and v2 in C(G) (which represents two faces in G) are connected by an edge if they are adjacent (have at least one edge in common) in G. Figure 2.2b and Figure 2.2c show an example for a cycle graph and the modified cycle graph respectively. We can see that node 5 is connected to both node 1 and node 4 since Face*5* is adjacent to Face*1* and adjacent to Face*4*.

[3] Cycle graphs are defined in the literature as the non-empty intersection between every two elements of the cycle basis set of a graph. Therefore they are also called the intersection graphs of the basis. The cycle graphs in [Sysl81] uses a special type of cycle basis called the plane cycle basis.

We use the following notations for the cycle graph:

N': the number of nodes in C(G), which equals the number of faces in G.

M'.: the number of edges in C(G).

D'_{avg}: the average nodal degree in C(G).

It can be shown that $N'=M-N+1$ using the observation that in any planar graph deleting edge will reduce the number of faces N' by one and the fact that we have $M=N-1$ in a tree.

It is interesting to notice that a cycle of faces in G is a simple cycle in C(G). An example can be found in Figure 2.2b later. We can see that node 1 is connected by an edge to node 2, Face*3* and Face*5* since the corresponding Face*1* shares edges with each of the faces Face*2*, Face*3* and Face*5* in G.

2.2.2 Modified Cycle Graph

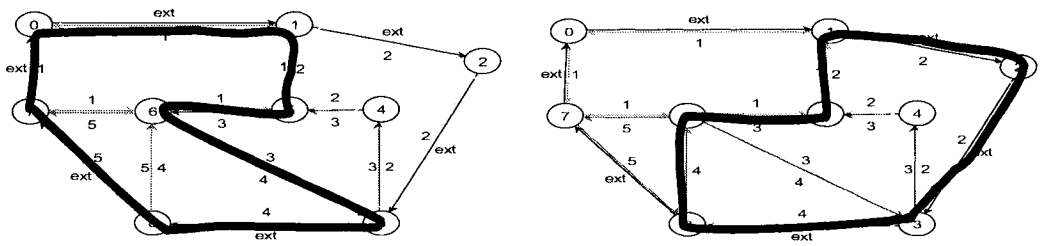
Although the regular cycle graph can capture the edge-jointed property between the faces of a planar graph, it cannot account for the node-jointed property and the number of edges shared between the faces. Therefore, we want to introduce the following modified cycle graph concept.

In the modified cycle-graph MC(G) of a graph G, the nodes are defined in the same way as for the regular cycle graph, i.e., they are the planar faces of G. However, in addition to showing the adjacency of two faces, we repeat an edge a number of times equal to the number of common (shared) edges between the two faces in G. This is the same idea as forming the geometric dual graph except that we do not consider the external face[4]. Furthermore, we shall use another set of edges, called the *dashed edges*, between the nodes in MC(G) when the two corresponding faces in G are node-jointed. Each dashed edge is given a label set consisting of the labels of all faces sharing the same common node in G (but not the two faces corresponding to the end-nodes of the dashed edge). For convenience, this label set has a similar notation as the label at a node in MC(G) except that when it appears at a node it means only one face while at a dashed edge it represents a set of faces.

As a consequence of our modification, we now need two more parameters for each node in the modified cycle graph MC(G): the parameter $e(n)$ which is the number of edges forming

[4] There are two categories of dual graphs: geometric dual graph and combinatorial dual graphs

the corresponding face in G, and the node degree *deg(n)* which is the number of regular (not dashed) edges incident at that node.

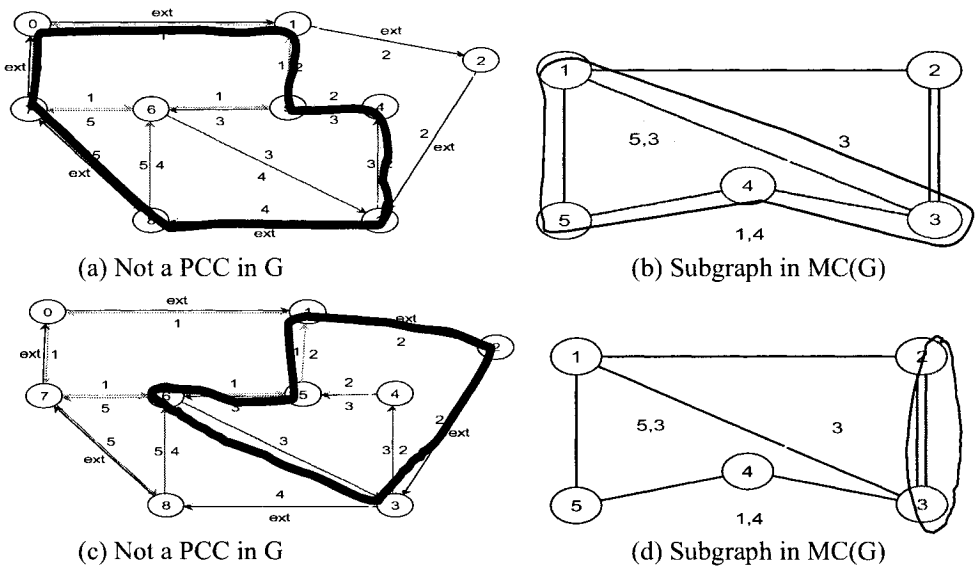


(a) PCC (b) Not a PCC

Figure 2.3: PCC Examples

2.3 PCC Model

One important type of cycle in planar graphs is called the PCC (*Planar Chordal Cycle*). This is defined as a simple cycle in a planar graph such that all the links inside the cycle are chords. Figure 2.3 shows an example of two simple cycles in G. From the definition Figure 2.3a is thus a PCC while Figure 2.3b is not. We shall characterize in the next chapter the existence of PCCs in planar graphs with respect to a set of faces.



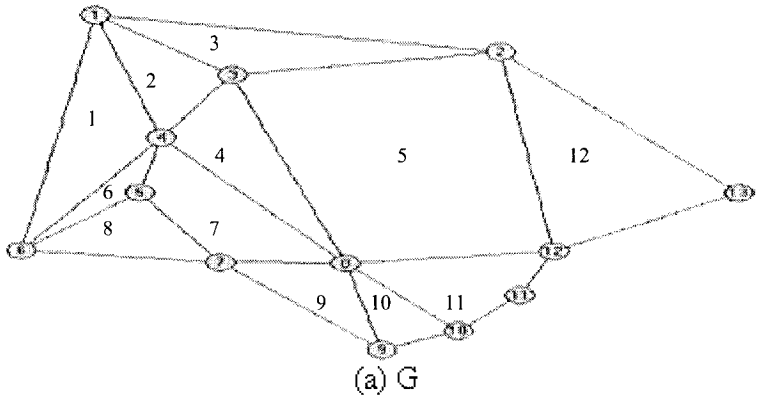
(a) Not a PCC in G (b) Subgraph in MC(G)

(c) Not a PCC in G (d) Subgraph in MC(G)

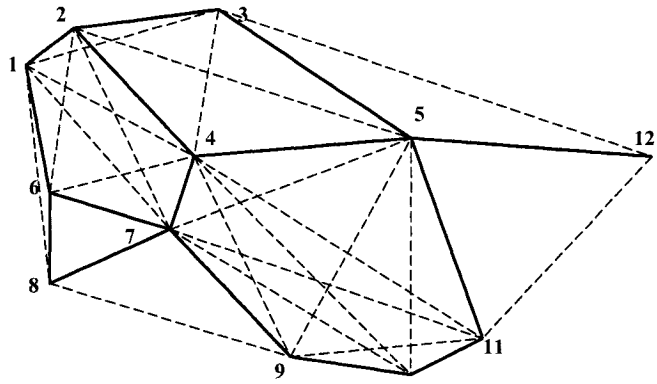
Figure 2.4: PCC in G and MCG

In general, any connected subgraph in the dual graph domain would correspond to a cycle (or a set of node-disjoint cycles) in the regular graph, but not always a PCC (or a set of PCCs). Figure 2.4 shows two examples (parts (b) and (d)) of subgraphs in the dual domain that do correspond to simple cycles (parts (a) and (c)) in G but are not PCCs. Our MCG

modification is indeed helpful to identify these cases. Figure 2.4b shows the subgraph (shaded area) that corresponds to the cycle shown in Figure 2.4a. As seen, the four faces sharing the same node in Figure 2.4a become the cycle 1-3-4-5-1 in Figure 2.4b. In the case of Figure 2.4d we have a "cycle" of two nodes which corresponds to two faces having two common edges Figure 2.4c. These observations will be generalized in the next chapter.



(a) G



(b) MC(G)

(labels of dashed edges not shown for simplicity)

Figure 2.5: Canada Network and its MCG

2.4 Examples

Figure 2.2 illustrates the difference between the cycle graph and its modified version. Node 2 and node 3 are connected with one edge in the regular cycle graph (Figure 2.2a) and by two edges in the modified cycle graph (Figure 2.2b). In addition, there are three dashed edges. One of them is between node 1 and node 4 because faces C1 and C4 are node-jointed. This common node is also a common node for faces C3 and C5; thus, "3" and "5" are included in the set of labels for this dashed edge. Moreover, Node 4 in Figure 2.2b has *deg*(4)=2 (i.e.,

node 4 in C(G) of Figure 2.2a has a node degree of 2) and e(4)=3 (i.e., face C4 in G is bounded by three edges)

In Figure 2.5 we show an example of a real network (Canada network) and its modified cycle graph. The network has 13 nodes and 24 edges. Notice that all the pairs of the adjacent faces have only one edge shared between them, thus the cycle graph and the modified cycle graph are the same except for the dashed edges.

2.5 Test Network Topologies

We present real network examples that will be used throughout the thesis in addition to some simple networks created. Another type is the random graph generated using the software package LEDA [LEDA06]. We present in the following a quick description of examples for those categories.

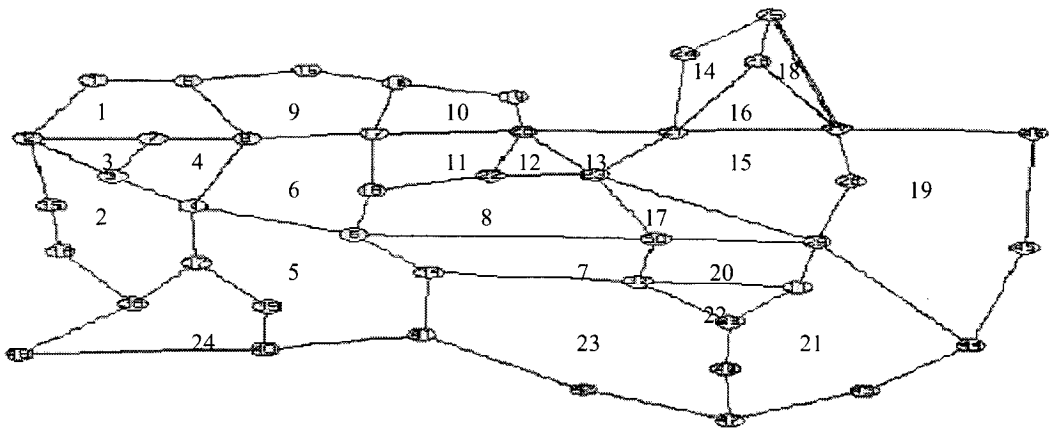


Figure 2.6: USA Long Haul Network

The first real network example is the one shown in the previous section in Figure 2.5, which represents the Canada network and has 13 nodes and 24 edges (nodal degree 48/13≈3.7). The other two real network examples are two versions of the USA long haul network. Figure 2.6 show the extended version of the USA long haul network with 42 nodes and 66 edges (nodal degree 132/42≈3.1). Figure 2.7 shows the simplified USA long haul network with 28 nodes and 45 edges (nodal degree 90/28≈3.2). The faces of both figures are given labels which will be used later on. Notice that the labels in Figure 2.6 are placed in the center of each face while the labels of the faces in Figure 2.7 are placed on the edges surrounding each face.

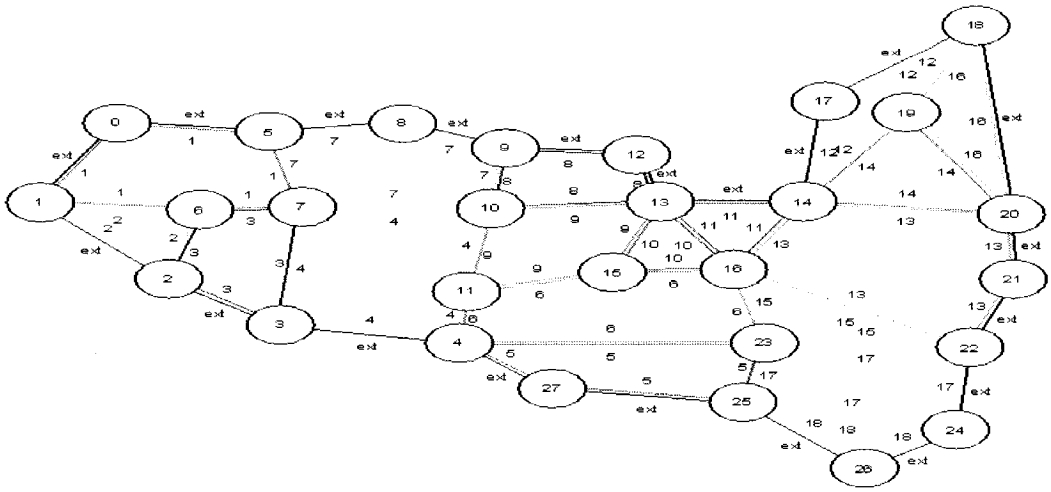
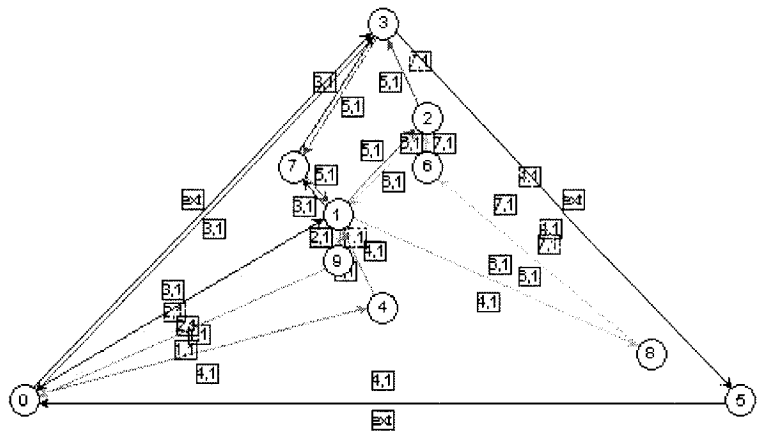
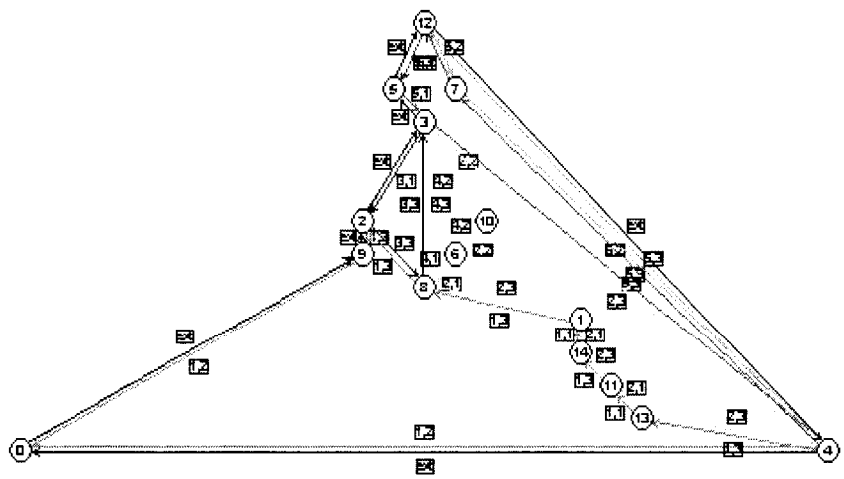


Figure 2.7: The Simplified USA Long Haul Network



(a) *N*=10, *M*=16



(b) *N*=15, *M*=20

Figure 2.8: Random Graphs Examples

The other type of graphs are the random graphs created by software using LEDA libraries. All the graphs are 2-connected graphs. A random topology is generated and the nodes are assigned labels from 0 to N-1 (where N is the number of nodes). Figure 2.8 shows 2 examples of different sizes created randomly. The labels on the edges shown in small squares are the face labels which have been created by an internal algorithm in the package.

2.6 Assumptions

The following assumptions hold for the reminder of the thesis unless otherwise stated:

- All edges are bidirectional unless explicitly stated.

- All graphs are planar unless otherwise stated.

2.7 Concluding Remarks

In this chapter, we have presented our network model in the regular and the dual domain with a set of definitions and terminologies. We have also presented an important modification for the dual graph representation which will capture the planar properties relating a new type of cycle called a PCC. In the next chapter, we shall discuss the properties of PCCs and the set of rules that is necessary and sufficient for their existence.

Chapter Three
PCC Properties and Generation Algorithms

Following the definitions and modifications of the cycle graph domain in the last chapter, we shall characterize the PCC in the regular and the dual domains. We then present different formulations for algorithms to generate PCCs. All these form the basis for various applications in future chapters.

3.1 PCC (Planar Chordal Cycle) Characterization

We start our discussion by making some observations on the characteristics of one, two, and three adjacent faces of a planar graph in forming a PCC. Then we will generalize the rules for forming a PCC from *n* adjacent faces. We will discuss them separately in two domains: the regular graph domain and the dual graph domain, in order to appreciate the correspondence.

3.1.1 PCC in Regular Domain

We study some simple cases of two and three faces before we generalize them. A complete list of all simple cases is provided in Appendix A for completeness.

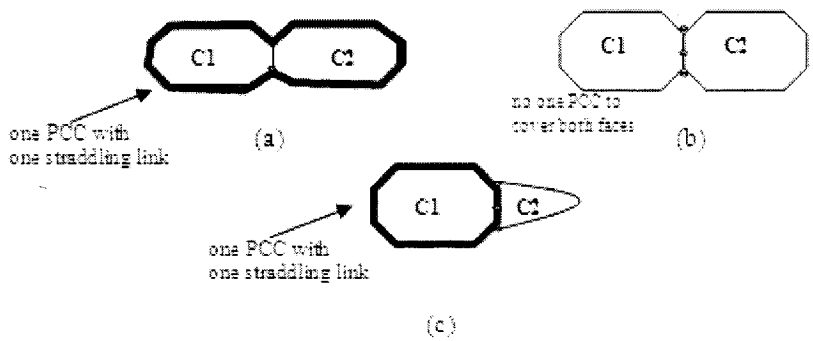


Figure 3.1: Different Cases of Two Adjacent Cycles

3.1.1.1 Simple Cases

Figure 3.1 depicts some examples of two faces. One can see that Cases a) and c) correspond to a PCC whereas Case b) does not because there are two edges shared between the two faces and these faces have many edges other than those shared ones.

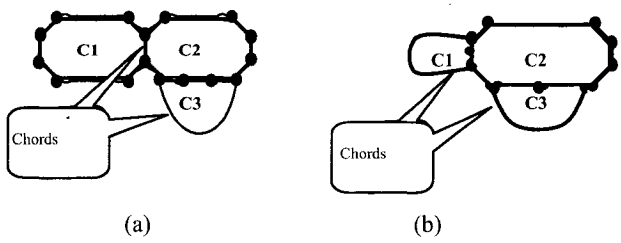


Figure 3.2: Example Cases of Three Adjacent Cycles

Since there are many cases of three adjacent faces, we show only a couple examples in Figure 3.2. A complete list is provided in Appendix A. Figure 3.2a is a case when two faces (C1 and C2) share one edge between them while the third one (C3) shares more than one edge with C2. In this case, we have a PCC formed by taking the two faces (C1 and C2) as internal faces for the PCC. Figure 3.2b shows that one face C2 can have more than one edge shared with each of the other two faces C1 and C3. In addition, each of these two faces has only one edge that is not shared with the first face C2. In such a case, a PCC is possible by taking only the edges of the first face (C2) as the cycle edges for the PCC. The PCCs in the figures are shown with bold edges.

Before we extend the idea to many faces, we identify two classes of PCCs. The first class is for PCCs whose on-cycle edges are not shared between adjacent faces. In such a class, all of the faces are *internal faces* for the PCC. Figure 3.1a is a typical example for two faces in which we see that all the "non-shared" edges of faces C1 and C2 are forming the PCC (i.e., C1 and C2 are internal faces for the PCC).

The other class is for PCCs that contain at least one external face with respect to the PCC that is covered by the PCC. A face is said to be covered by a PCC when each edge of the face is either an on-cycle or straddling for the PCC. Figure 3.1c is the first example of this class. Figure 3.2 above has two examples: The PCC example in Figure 3.2a has two internal faces and one external, while in Figure 3.2b, there is one internal face and two faces external to the PCC.

3.1.1.2 Generalization

When there are more than 3 faces, we could encounter a problem when two internal faces of the PCC can be node-jointed and no neighboring faces are inside the PCC which will result

in a non-simple PCC. For example, in Figure 3.3b, faces C1 and C3 are two internal faces for the PCC traced out by the thick lines; they are node-jointed and the third face (C2), sharing the same node with these two internal faces C1 and C3, is an external face. Figure 3.3c shows the case where the three faces (C1, C2, and C3) are all internal faces and thus the cycle is considered a PCC.

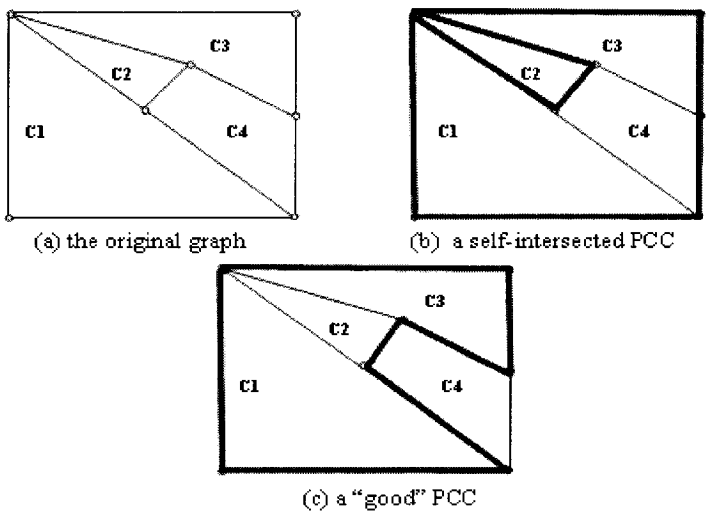


Figure 3.3: Face Adjacency at a Node and its Effect on the PCC

Given a set of faces and their edges in G, a PCC can be formed from the on-cycle edges or chords of internal and external faces provided that

(i) There is exactly one common edge between each pair of adjacent internal faces.

(ii) Any subset of the internal faces could not form a cycle of faces.

(iii) Each edge of the external face edges should be either a common edge with an internal face, or an edge joining two nodes of the internal faces nodes.

(iv) When two internal faces of the PCC are node-jointed at a node v which is not part of a common edge between these two faces, we require all the faces sharing this node to be internal faces.

Figure 3.4: PCC Conditions in Regular Domain

In general, for a set of k faces numbered from 1 to k, any face could be edge-jointed with any number of other faces (1, 2 ... up to k-1). Continuing with the same observations and deduction on two and three faces, we can now make the following statements on the conditions for forming one PCC from k faces. Alternatively, we can say that a PCC can be formed if the internal and the external faces of this PCC satisfy the conditions in Figure 3.4.

3.1.2 PCC in Dual Domain

We shall now describe the same observations about two and three adjacent faces in Section 3.1.1 using the modified cycle graphs. The numbers at the nodes are *e(v)* as discussed in Chapter Two (number of edges forming face *v* in G).

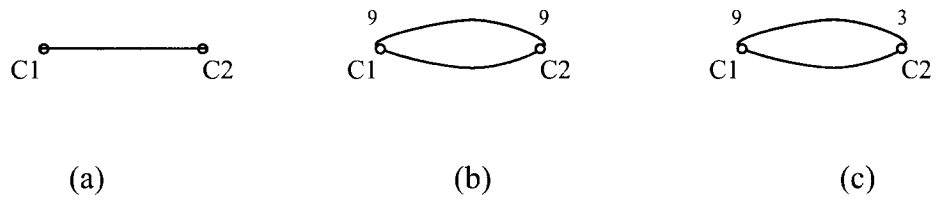


Figure 3.5: Cases of Two Adjacent Faces Represented in the Modified Cycle-Graph

3.1.2.1 Simple Cases

The cases of two and three faces in the dual domain can be classified and summarized by the following:

1) For two connected nodes in MC(G) as in Figure 3.5, we consider the following two subcases:
   a) When the two nodes are connected by only one edge as in Figure 3.5a (which corresponds to the case of two adjacent faces with one edge shared between them, as shown in Figure 3.1a), then there is a PCC to cover the edges of the two corresponding faces.
   b) When the two nodes are connected by more than one link (which corresponds to the case of two adjacent faces with more than one edge shared between them), then we will have a PCC to cover the links of the two corresponding faces either when $e(C1)=deg(C1)+1$ or $e(C2)=deg(C2)+1$ but not both (i.e., one of the faces has only one other link not common with the other face). For example, Figure 3.5b has two links (as it corresponds to Figure 3.1b), However, its $e(C1)=e(C2)=9$ and $deg(C1)=deg(C2)=2$, meaning that neither the relationship $e(C1)=deg(C1)+1$ nor $e(C2)=deg(C2)+1$ is satisfied. Now one would find that Figure 3.1c would form a PCC because as shown in Figure 3.5c $e(C2)=deg(C2)+1$. Recall that $e(v)$ is the number of edges Face *v* has in G; this is shown in Figure 3.5 and Figure 3.6 as numbers beside the nodes.

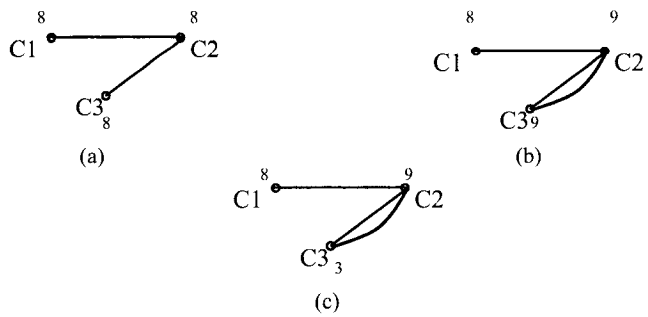


Figure 3.6: Cases of Three Adjacent Faces Represented in MCG (*e*(*N*) is Shown for Each Node.)

2) For the three-connected nodes in MC(G) as shown in Figure 3.6, there are also two sub-cases:
   a) When there is no cycle of three nodes in MC(G), a PCC can be found in G under either of the following conditions:
      i) If there is exactly one edge between (C1, C2) and between (C2, C3). See Figure 3.6a as an example.
      ii) If there is more than one edge between (C2, C3), but only one edge between (C1, C2) and $e(C3) = deg(C3)+1$, then a PCC also exists. In Figure 3.6b this does not hold and thus no PCC is possible. On the other hand, this relationship holds in Figure 3.6c and thus a PCC is possible as seen in Figure 3.2a.
   b) When there is a cycle for the three nodes in MC(G), a PCC can be found in G if at least one of the nodes (say v) in this cycle possesses the property that $e(v)=deg(v)+1$. For example, consider Figure 3.2a where imagine the edge that is hanging on two nodes in C2 to form C3 is hanging between a node in C1 and a node in C2. Note that the two nodes (here C1 and C2) other than v (here C3) should be under either Condition 1a or Condition 1b stated above.

3.1.2.2 Generalization

Similar to the generalization in 3.1.1.2 for the set of faces in G, the previous observations can be generalized to any number of nodes in MC(G). The following conditions in Figure 3.7 are for a set of nodes S in MC(G) that can guarantee the existence of a PCC in G in order to cover all edges of the corresponding faces.

Given a set of nodes S in MC(G), a PCC can be found to contain (or cover) all the edges of the corresponding faces in G if all the following conditions are met:

1) S can be divided into S_i and S_e such that the induced subgraphs I_S on S and I_{Si} on S_i (but not necessarily I_{Se} on S_e) are connected.
2) A set S_i must satisfy the following conditions where applicable:
   a) I_{Si} does not have cycles or multiple edges (considering only the regular not the dashed edges)
   b) I_{Si} can have a cycle from the regular edges with only one dashed edge and the label of this dashed edge includes the labels of all the nodes in the cycle except the two labels of its own end-nodes.
3) A set S_e contains only nodes such that a node member v observes the relationship $e(v)$-$deg(v,S) \leq 1$ where $e(v)$ is the number of edges forming face v in G, and $deg(v, S)$ is the number of nodes in the set S connected with node v.

Figure 3.7: Conditions for Finding a PCC in Dual Domain

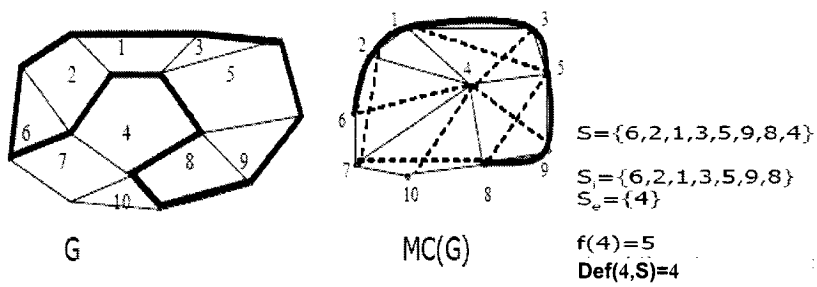


Figure 3.8: Example on the PCC Conditions

Figure 3.8 shows an example for the set S and a possible partition into S_i and S_e where the conditions in the dual domain are met and the PCC as illustrated.

3.1.2.3 Example When $S=S_i$

One particular scenario is when all the nodes of S belong to S_i.(thus $S_e=\varnothing$). Here, Condition 1 is satisfied since the induced subgraph on S_i is connected. Condition 3 is not applicable since no external nodes (faces) are assumed to exist. Therefore, we shall discuss Condition 2 using the following two examples to explain the applicability of the two subcases. Figure 3.9a is an example for Condition 2a to verify the existence of a PCC (there is no need to check Condition 2b because no dashed edges exist). It is a simple path from v_1 to v_k in MC(G).There are no multiple edges along this path, and no cycles formed from the

regular edges. Figure 3.9b shows the corresponding set of adjacent faces in G, and Figure 3.9c the PCC covering the edges of all these faces.

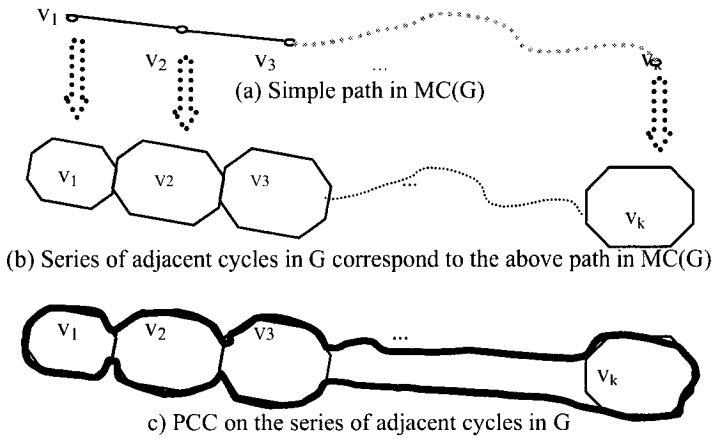


Figure 3.9: Set of Adjacent Faces Corresponding to a PCC (Without Dashed Edge)

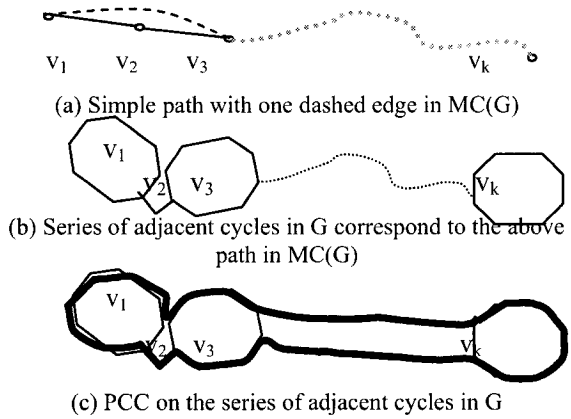


Figure 3.10: Set of Adjacent Faces Corresponding to a PCC (With Dashed Edge)

Figure 3.10a is an example for Condition 2b. It is a simple path with one dashed edge joining v_1 and v_3 for which we have to verify for the existence of a PCC. Since the label of this dashed edge includes v_2, then this dashed edge satisfies Condition 2b and the PCC can be formed as in the previous case by including all the faces as internal faces. Figure 3.10b shows the corresponding set of adjacent faces in G, and Figure 3.10c the PCC covering the edges of all these faces.

3.1.2.4 The Mapping Theorem

Based on the previous discussion, we provide the following theorem that prove the completeness and correctness the general mapping from G to MC(G) that would guarantee the existence of a PCC.

Theorem 1

Any connected subgraph[5], induced on a set S of nodes in MC(G), satisfying the conditions in Figure 3.7 will have at least one PCC that would cover the edges of all corresponding faces in G. A PCC can then be formed by taking the set of faces that correspond to the nodes in the set S_i as its internal faces.

Proof:

We shall use mathematical induction on the subset S having f nodes. Let N be the number of nodes in a given MC(G).

Boundary case of f=1

For f=1 node we see that all conditions in Figure 3.7 are satisfied, and there is only one simple PCC without any chordal links.

General case of $N > f > 1$

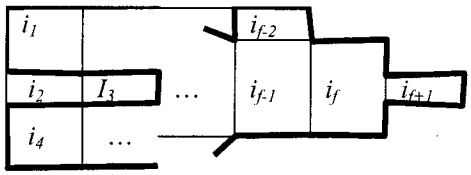
Assuming there exists a subset having $f > 1$ nodes that satisfy the conditions in Figure 3.7, (i.e., there exists a PCC in G covering all the links of the corresponding f faces), we want to show that a PCC can also be found on an induced subset with $f+1$ nodes, if such subset can be generated. In other words, we want to determine if an additional node can be a member of either set S_i or S_e while satisfying the conditions in Figure 3.7.

To proceed, remember that an external face can also be covered by a PCC (such as Face i_2 in Figure 3.11a). Figure 3.11b shows the corresponding MC(G) where the bold edges represents the subgraph of the internal faces of the PCC. According to the conditions in Figure 3.7, the f nodes can be divided into two sets S_i and S_e such that S_i contains the nodes corresponding to the internal faces and S_e contains nodes corresponding to the external but covered faces. Let k be the node degree of node i_{f+1}. Then there are several cases to consider

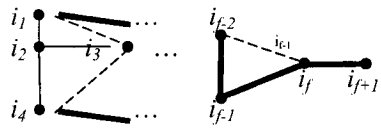
---

[5] We consider whether the subgraph on the regular edges is connected or not without considering the dashed edges.

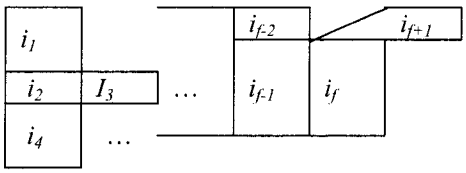
in which a new node i_{f+1} (a face in G) can be connected to the other nodes (faces).



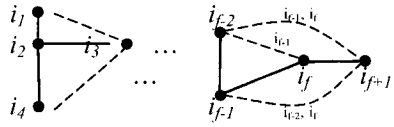
a) New face has only one common edge with the existing faces (case 1)



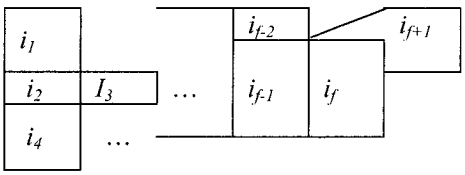
b) Equivalence of (a) in MC(G)



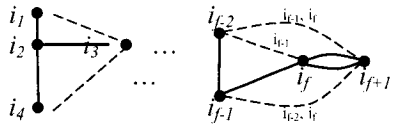
c) New face has only one common edge with the existing faces (case 1)



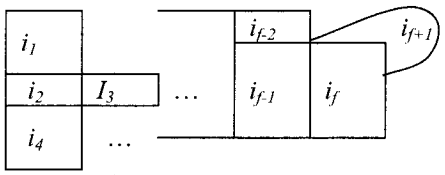
d) Equivalence of (c) in MC(G)



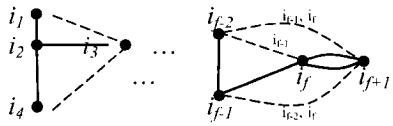
e) New face has two common edges with the existing faces (case 2).



f) Equivalence of (e) in MC(G)



g) New face has two common edges with the existing faces (case 2)



h) Equivalence of (g) in MC(G)

Figure 3.11: Different Cases for a New Face Joining the Existing Set of Faces

1) Case of k=1:

In this case we have only one face that is edge-jointed to the Face i_{f+1}, i.e., there is one regular edge joining i_{f+1} to another node. Regarding the dashed edges, we have two cases:

1a) No dashed edges are connected to i_{f+1} (e.g., Figure 3.11b): In this case, node i_{f+1} is an additional node in set S_i and is connected to only one member of the same set. So no cycle of any kind exists, and Condition 2a in Figure 3.7 is satisfied. The PCC of an example in such a case is shown in Figure 3.11a.

1b) A dashed edge is connected to i_{f+1} (e.g., Figure 3.11d): There are two more sub-cases depending on whether the dashed edge would form a cycle with the bold edges or not:

i) Not forming a cycle e.g., edge (i_1,i_3): In this case, there is no cycle violating Condition 2b.

ii) Forming a cycle (e.g., edge (i_{f-1},i_{f+1})) with the bold edges (i_{f-1},i_f) and (i_f,i_{f+1})): In this case, the label i_f is included in the label of this dashed edge, and then Condition 2b is satisfied.

One can see that in both cases, there is no violation of Condition 2a by the assumption of the mathematical induction. Since the node i_{f+1} is assumed to be a member of the set S_i, then Condition 3 does not change and still satisfied by the induction hypothesis since it deal with S_e and nothing change to this set. Therefore, a PCC can be formed by adding an extra face as shown in Figure 3.11c.

2) Case of k>1:

Under this case, whenever node i_{f+1} satisfies the relationship of $e(i_{f+1})$ - $deg(i_{f+1},S)$ = 1, one finds that the existing PCC already covering the links of the first f faces would also cover Face i_{f+1} (as an external face to that PCC). This is because there is only one edge for Face i_{f+1} that is not already one of the on-cycle edges and this edge is a chordal link for the existing PCC. Figure 3.11g and Figure 3.11h illustrate the case where this condition is satisfied. $deg(v, S)$ represents degree node v among the subset set S (i.e. the number of edges connects v to nodes in set S). On the other hand, if the new face dose not satisfy the condition, i.e., if $e(i_{f+1})$ - $deg(i_{f+1}, S) \neq 1$, then the PCC does not exists, as illustrated in Figure 3.11e and Figure 3.11f.

In summary, a PCC can be found for $f+1$ nodes whenever the conditions of Figure 3.7 holds. We therefore complete the proof by mathematical induction.

Corollary 1:

If the conditions of Figure 3.7 are satisfied on the whole graph MC(G) (i.e. all the nodes are either in S_i or S_e), then all the links in the original graph can be covered by one PCC, and this PCC is a Hamiltonian Cycle of the graph G.

Proof:

According to Theorem 1, all edges (within a subgraph of the network) that satisfy the

conditions in Figure 3.7 are covered by one PCC. Since each covered edge is either an on-cycle edge or a chordal link, their end points must lie on the cycle. By extending the theorem to the whole network graph, it follows that all nodes are covered by one cycle which is the definition of a Hamiltonian cycle.

3.2 Algorithms in Dual Domain to Generate PCCs

After the proof of the sufficient conditions, we are now in a position to provide a simple algorithm to build the modified cycle graph from a given graph G. Then we want to present two more common algorithms for various applications to be used in later chapters. Before we do so, we shall define the following terminologies.

3.2.1 Terminology and Definitions

deg(v, S): the number of edges joining node v with nodes in the set S.

"non-visited": the initial state of the nodes before any iteration of the algorithm.

"visited": the state when a node is included in the final subgraph(s), such that the same nodes in the same connected subgraph in MC(G) corresponds to one PCC in the original graph.

"blocked": the state when a node will not be included in the final subgraph(s), because such nodes will prevent a PCC from forming with its visited neighbor nodes.

"covered": the state when a blocked node has cover number less than 2.

Likewise, we define a set of nodes to be Non-visited, Visited, Blocked, or Covered according to the corresponding state the set is in.

Let $c(v)$ be the "cover number" of node v which is the number of edges the corresponding Face v has in G minus the degree of node v on the set "visited". In other words, $c(v) = e(v) - deg(v, Visited \cup Covered)$ where U is the union set operator. This variable $c(v)$ determines how many edges for the corresponding Face v in G are not covered with the PCC that formed in the Visited set. The other faces corresponding to the set of nodes in the set Covered will be external faces to this PCC and covered by it.

By calculating $c(v)$ for each node in the blocked set, one can form a new set called Covered as defined above. That is, Covered = $\{x | x \in$ Blocked and $c(x) \leq 1\}$). In the beginning of the

algorithm, all nodes are initialized to the "Non-visited" state, but at the end of the algorithm, they would belong to one of the 4 sets of nodes called "Non-visited", "Visited", "Blocked" and "Covered". Note that the set Visited and Covered correspond respectively to the sets S_i and S_e already explained in Section 3.1.2.2. The PCC in G can now be formed by taking the faces corresponding to all nodes in the set Visited as internal faces. The faces corresponding to the nodes in the set Covered will be external faces to this PCC.

3.2.2 The MCG Building Algorithm

Figure 3.12 gives the pseudo code for building the modified cycle graph. The initialization numbers the faces one by one by uniquely representing each face through its bounding edges. Consequently, a graph G can be represented by a list of faces which in turn is represented by a list of edges, all of which are stored in the database of a program.

To build the MC(G) of a graph G, we create a node n_i corresponding to each face f_i in G, and then traverse the edges of the face f_i edge-list to see if an edge in the list also appears in the list of any other face f_j list. If so, connect the node n_i with the node n_j.

The procedure is similar in establishing the dashed edges, but one would examine the end nodes of each edge (rather than the edge itself) in a face-list. Nodes n_i and n_j are connected by a dashed edge if an end node of an edge in the edge-list of face f_i also appears as an end node in the edge-list of face f_j except if the edge itself appears in both lists.

The above algorithm is summarized in Figure 3.12, and implemented in C/C++ [DeDe98]. The LEDA libraries [LEDA] are used to generate graphically the modified cycle graph for a given graph G. The "Sense of direction" is defined to be the awareness of each edge for the edges around its end points and their order.

The complexity of the above algorithm is $O(M^3\text{-}M^2N)$ since the number of faces is $O(M\text{-}N)$ and each face list is bounded by M edges and the face list is compared to each other face (i.e. we have $(M\text{-}N)^2M = M^3\text{-}M^2N$).

Figure 3.13 is an example generated by the algorithm. This is a real network called USA long haul network with 28 nodes and 45 edges (i.e. nodal degree $90/28 \approx 3.2$) as shown in Figure 3.13a. Figure 3.13b is the corresponding MCG graph having 18 nodes and 24 regular edges

and 22 dashed edges. The details of such mapping has also been previously exemplified in Section 2.4. Note that the labels for the dashed edges are hidden for better viewing.

```
Alg_MCG (G)  // Applicable for a planar graph G
Begin Alg_MCG
        Initialization: Assign a unique number for each face in the graph
        G and get the list of edges bounding each face.
            To build this list we should have kind of "sense of directions" for the edges
            intersection at each node. In other words, we need an ordering of the edges in clockwise
            or counter-clockwise direction at each node starting at any edge. Now starting at any
            node you can list all the faces around that node. We can do that for all nodes, then all
            faces will be listed.
        For each face f_i in G: Add a node n_fi in the graph MC(G)
        For each face f_i in G
          For all edges k in f_i
               If k appears in f_j list
                     Add a regular edge between n_fi and n_fj in MC(G)
               If an end point of k appears in face f_i but not k
                     Add a dashed edge between n_fi and n_fk in MC(G) with a
                     label consisting all faces have the end point of k in
                     their list.
End Alg MCG
```

Figure 3.12: Pseudo-Code for MCG (Modified Cycle Graph) Algorithm

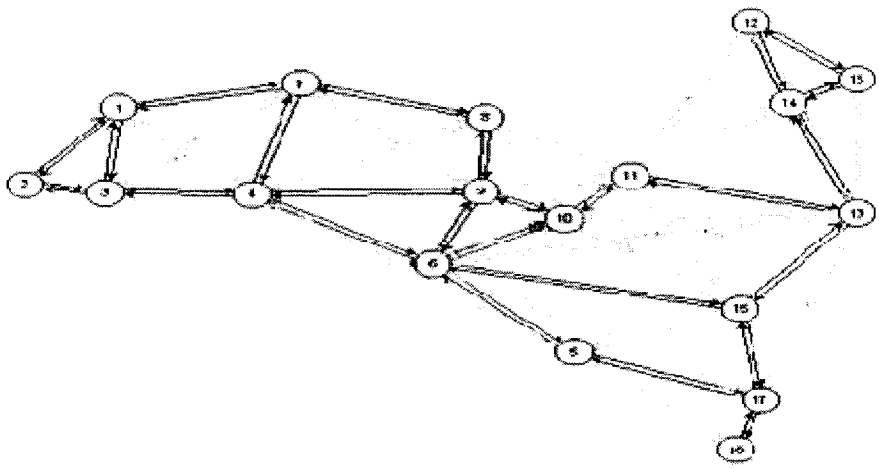


Figure 3.13: MCG for "Simplified" USA Long Haul (Figure 2.7)

3.2.3 Breadth First Search-Based Algorithm

Given a starting node in MC(G), the fundamental algorithm in the previous section did not specify the way/order how other nodes are explored/searched in order to form a PCC. We propose here the Breadth-First Search as a common approach we use in our research and later we present a Depth-First Search method.

In general, the BFS algorithm searches all nodes that are k-hop away from a starting node before nodes that are $(k+1)$-hops away. We use a queue to implement our BFS algorithm. We start by adding the neighbor nodes of the starting node to the queue and iteratively remove one node from the head of the queue. We search for the neighbors of this dequeued node and add/enqueue any of its "non-visited" neighbors to the end of the queue. For each enqueued node, a tree edge is marked after joining this node with its proceeding (parent) node. The resulted tree is called BFS tree.

There are a few conditions to check according to Figure 3.7 and corresponding actions taken before enqueuing the node:

1) Calculate and check if e(v)-*deg*(v,Visited) $\leq$1. If so, Condition 3 is satisfied, mark those nodes "covered"; otherwise mark as "blocked"

2) Check if either Condition 2a or 2b is met by the following procedure:

   Condition 2a: look one step further to see if any nodes there are already visited. If so, mark it as "blocked"; if not, mark it as "visited".

Condition 2b: check whether the present node being enqueued has a dashed edge with an already-"visited" node. If so, compare the labels and mark accordingly as visited or blocked, and if not, mark it as "visited".

Figure 3.14 is the pseudo code of this BFS algorithm which is called the PCMCG (Planar Cycles using MCG). Here the Q.Push adds a node to the end of the queue Q while Q.pop removes a node from the front of the queue. The instruction "find_cycle(visited)" finds the cycle in the original graph that correspond to the nodes marked as visited in the MC(G) graph. We also use two functions to check the cycles found: *cycle_regular* and *cycle_dashed.* The function *cycle_regular* checks if the given nodes forms a cycle or not on the regular edges whereas cycle_dashed checks if the given nodes forms a cycle with one dashed edges

and other regular edges from the given nodes. The matrix variable A' represents the adjacency matrix for the modified cycle graph.

```
Initialize array status[] of size equal to the number of nodes in MC(G) to not-visited.
Call Alg_PCMCG(v) // calls Alg_PCMCG to mark the nodes that can form a PCC
                  // with v as visited
PCC=find_cycle(status) // generates the PCC based on the resulted status
                       // value
```

```
Alg_PCMCG (v)
Begin Alg_PCMCG
        Status[v]=visited
        Q.Push(v)
        while Q is not empty do
                v=Q.pop
                get A'(v) and organize list of nodes (y) based on some ordering
                for all nodes y ∈ A'(v)
                        if node y is not visited then
                                if (cycle_regular(v,y,z) and status[z]=1 from some node z)
                                OR (node v and node y connected by multiple edges) then
                                status[y]= blocked,
                        else If (cycle_dashed(x,y,z) and labels(y) ∈ cycle labels)
                             OR (no edge between node v and node y)
                             status[y]= visited
                             Q.push(y)
                        For all nodes y ∈ A'(v)
                                If status[y]= blocked and c(y) ≤1 then status[y]= covered
End Alg_PCMCG
```

Figure 3.14: Pseudo-Code for PCMCG (PCC in MC(G)) Algorithm Using BFS

The BFS algorithm is illustrated by a step-by-step evolution from of the original graph (Figure 3.15) and the corresponding MC(G) in Figure 3.16. Notice how the algorithm starts from Face 3 (shaded in Step 1) and considers all of the adjacent faces (one-hop neighbor) before moving to a new face that is 2-hops away. The neighbors in our algorithm are scanned in ascending order of their face indices.

For example, the algorithm chooses its first neighbor, Face 2, (Faces 2 and 3 are now shaded in Step 2). The algorithm then chooses Face 4 as the next neighbor (Step 3), which will be marked as "blocked" because it has multiple edges with Face 3. Finally, Face 5 will be considered as the last neighbor of Face 3 (in Step 4). After all of the three neighbors have been considered, the algorithm moves back to the lowest-labeled neighbor of Face 3,

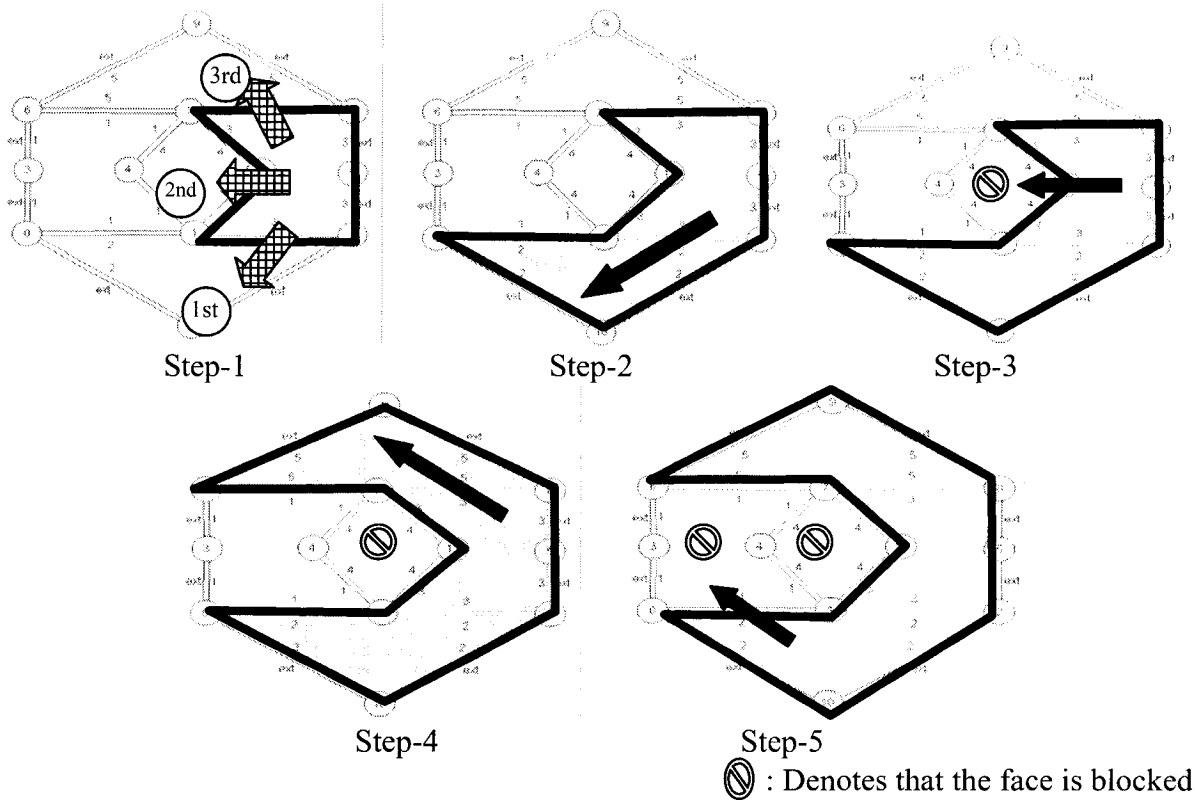


Figure 3.15: Using BFS Option for the MCG Algorithm Shown in the Original Graph

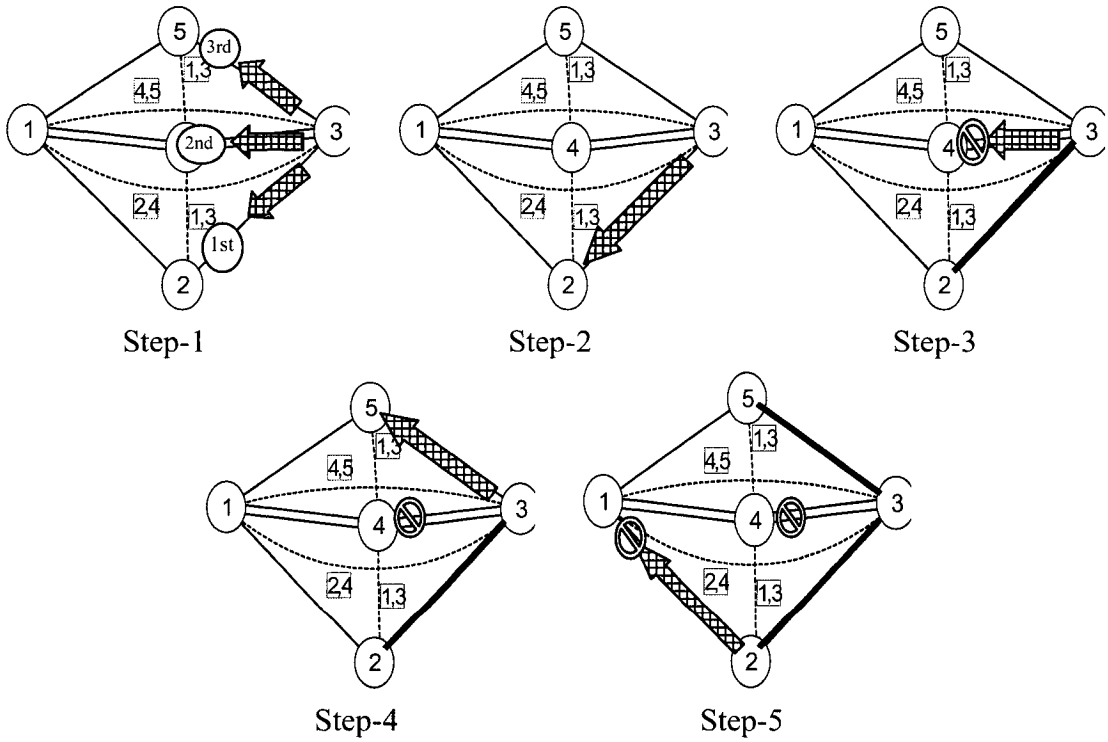


Figure 3.16: Showing Figure 3.15 in the MCG Domain

and considers its neighbors there, effectively considering neighbors that are two-hops away. In this case, Face 2 is the lowest labeled neighbor of Face 3, and the algorithm chooses Face 1(Step 5) as the two-hop neighbor. Face 1 is marked as "blocked" because it is forming a cycle with Face 2, Face 3, and Face 5.

The complexity of the algorithm is $O(N'M'^2)$ since each node can be in the queue at most once and each neighbor for each node being examined is visited and the neighbors of the neighbors are visited for verifications at each step. where N' and M' are defined in Section 2.2. Since $N'= M-N+1$ and $M'<M$, the complexity of the algorithm is therefore $O(N'M'^2)= O((M-N+1).M)= O(M^2-MN)$

```
Initialize array status[] of size equal to the number of nodes in MC(G) to not-visited.
Call Alg_PCMCG2(v) // calls Alg_PCMCG2 to mark the nodes that can form a PCC with v
                   // as visited
PCC=find_cycle(status) // generates the PCC based on the resulted status value
```

```
Alg_PCMCG (v)  // first call with input node v
Begin Alg_PCMCG2
               Status[v]=visited
               Q.Push(v)
       while Q is not empty do
               v=Q.pop_front
               get A'(v) and organize list of nodes (y) based on some ordering
               For all nodes y ∈ A'(v)
                    if y is not visited then
                        if (cycle_regular(v,y,z) and status[z]=visited for some node z )
                        OR (node v and node y connected by multiple edges) then
                                    status[y]= blocked,
                        else If(cycle_dashed(v,y,z) and labels(y) ∈ cycle labels)
                              OR (no edge between node v and node y)
                          status[y]= visited
                          Q.push_front(y)

               For all node y ∈ A'(v)
                      If status[y]= blocked and c(y) ≤1 then status[y]= covered
   End Alg_PCMCG2
```

Figure 3.17: Pseudo-Code of PCMCG Using DFS

3.2.4 Depth First Search-Based Algorithm

The DFS is essentially the same algorithm as the breadth first search except for the modified node traversing behavior. It is an alternative method to the BFS. Unlike the BFS, the DFS picks one neighbor node and continues to search from that neighbor node in increasing hop-

distance from the starting node. The pseudo code for this implementation is shown in Figure 3.17. Here Q.pop_front and Q.push_front, respectively, removes and adds a node from the front of the queue.

Table 3-1: Results of Applying PCMCG on the Graph of Figure 2.5

Starting face :	PCC faces:	total edges/ chords + cycle edges	On cycle edges	Chords edges
0	1,6,2,4,8	24/13	(1,2),(2,3),(3,8),(8,4),(4,5),(5,7),(7,6),(6,1)	(1,3),(1,4),(3,4),(4,6) ,(5,6)
1	0,2,4,6,8	24/13	(1,2),(2,3),(3,8),(8,4),(4,5),(5,7),(7,6),(6,1)	(1,3),(1,4),(3,4),(4,6) ,(5,6)
2	1,5,0,11,12,6,10,8	24/24	(1,2),(2,13),(13,12),(12,11),(11,10),(10,9),(9,8), (8,3),(3,4),(4,5),(5,7),(7,6),(6,1)	(1,3),(1,4),(2,3),(2,12), (4,6),(4,8),(5,6),(7,8),(7,9),(8,10), (8,12)
3	No such face	24/0		
4	1,5,7,0,11,12,9	24/24	(1,3),(3,2),(2,13),(13,12),(12,11),(11,10),(10,8), (8,9),(9,7),(7,5),(5,4),(4,6),(6,1)	(1,2),(1,4),(2,12),(3,4),(3,8),(4,8), (5,6),(6,7),(7,8),(8,12),(9,10)
5	2,4,11,12,7,10,6	24/24	(1,2),(2,13),(13,12),(12,11),(11,10),(10,9),(9,8), (8,7),(7,5),(5,6),(6,4),(4,3),(3,1)	(1,4),(1,6),(2,3),(2,12),(3,8),(4,5), (4,8),(6,7),(7,9),(8,10),(8,12)
6	0,7,1,9,2,10,11	24/22	(1,2),(2,3),(3,4),(4,8),(8,12),(12,11),(11,10), (10,9),(9,7),(7,5),(5,6),(6,1)	(1,3),(1,4),(2,12),(3,8),(4,5),(4,6), (6,7),(7,8),(8,9),(8,10)
7	4,6,9,1,5,10,12	24/22	(1,3),(3,2),(2,13),(13,12),(12,8),(8,10),(10,9), (9,7),(7,5),(5,6),(6,4),(4,1)	(1,2),(1,6),(2,12),(3,4),(3,8),(4,5), (4,8),(6,1),(6,7),(7,8),(8,9)
8	6,0,1,2,4	24/14	(1,2),(2,3),(3,8),(8,4),(4,5),(5,7),(7,6),(6,1)	(1,3),(1,4),(3,4),(4,6),(5,6),(7,8)
9	7,10,4,6,11,1,2	24/22	(1,2),(2,3),(3,8),(8,12),(12,11),(11,10),(10,9), (9,7),(7,5),(5,6),(6,4),(4,1)	(1,3),(1,6),(2,12),(3,4),(4,5),(4,8), (6,1),(6,7),(7,8),(8,9),(8,10)
10	9,11,7,5,6,2,12	24/24	(1,2),(2,13),(13,12),(12,11),(11,10),(10,9),(9,7), (7,5),(5,6),(6,4),(4,8),(8,3),(3,1)	(1,4),(1,6),(2,3),(2,12),(3,4),(4,5), (6,7),(7,8),(8,9),(8,10),(8,12)
11	5,10,2,4,12,9	24/18	(1,2),(2,13),(13,12),(12,11),(11,10),(10,9),(9,7), (7,8)(8,4),(4,3),(3,1)	(1,4),(2,3),(2,12),(3,8),(8,9),(8,10), (8,12)
12	5,2,4,11,7,10,6	24/24	(1,2),(2,13),(13,12),(12,11),(11,10),(10,9),(9,8), (8,7),(7,5),(5,6),(6,4),(4,3),(3,1)	(1,4),(1,6),(2,3),(2,12),(3,8),(4,5), (4,8),(6,7),(7,9),(8,10),(8,12)

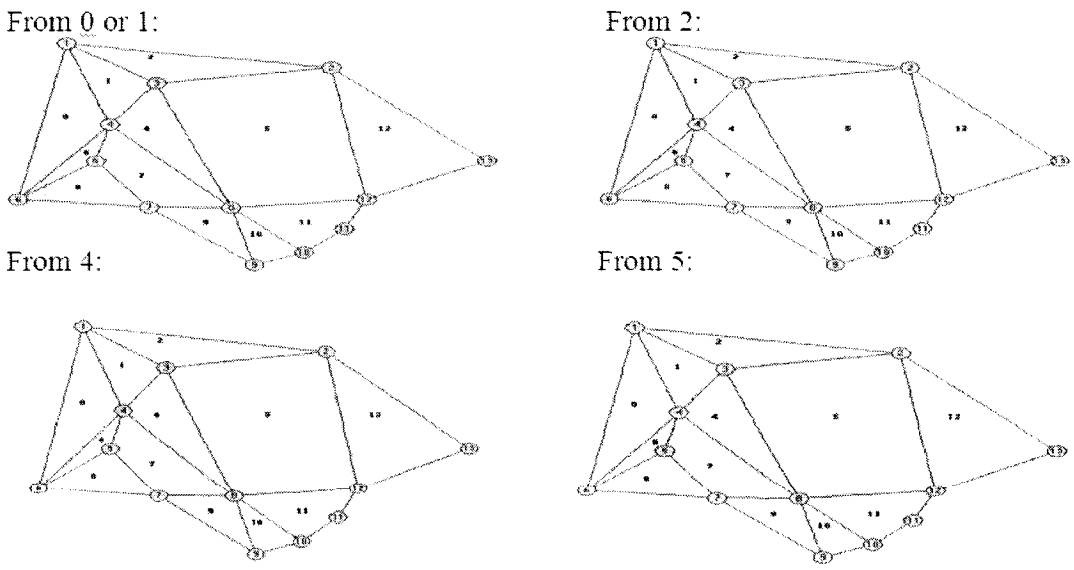


Figure 3.18: Illustration of some PCCs from Table 3-1

3.2.5 Example: PCCs Using BFS

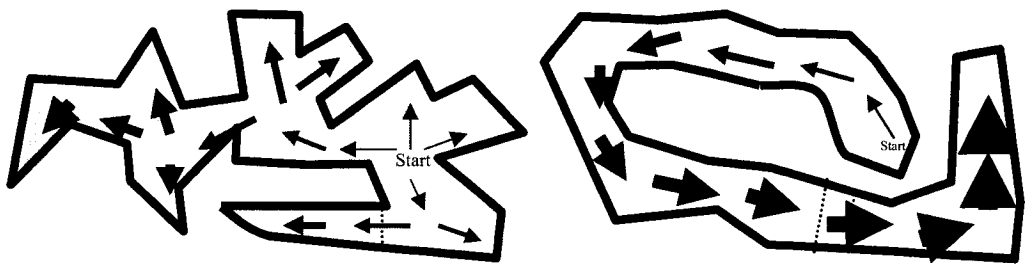
Here we show an example in which we have generated all the possible PCCs using the BFS technique. The network graph was shown in Figure 2.5. The results in Table 3-1 show the maximum PCC generated starting from each face. The PCC internal faces are shown in column 2 of Table 3-1. Column 3 shows the ratio of the total number of edges to the total number of on-cycle and chord links which are shown, respectively, in columns 4 and 5. We can see that a Hamiltonian cycle was obtained in different cases where we have 24/24 which means that 24 edges (all of them) in that cycle either are on-cycle or chords. Some of the results shown in the table are shown graphically in Figure 3.18.

3.3 Cycle Properties

In general, the number of all simple cycles in a graph increases exponentially with the number of nodes and edges. Our algorithms introduced above would cut down the tedious task of enumerating these cycles which is an NP-complete problem. In addition, our algorithms may generate cycles of different properties and therefore each type of algorithm may lead to a different application. We shall illustrate this by first comparing BFS with DFS before discussing other desirable properties.

3.3.1 BFS vs. DFS Cycles

Even though BFS and DFS could result in similar cycles in very special cases in which the ordering of the edges and nodes enforces choosing the same next node (e.g. see section 3.1.2.3). Some observations and properties are made below.



(a) BFS (b) DFS

Figure 3.19: Comparing BFS and DFS PCCs

Since the BFS and DFS search the graph space in different ways, we can observe different general properties of the PCCs they generate as demonstrated in Figure 3.19. The shaded area represents links that are chords or on-cycle while those outside might not be. The dotted edge represents an example of one chord (in general we could have many). We can have the

following general observations:

1) The shorter of the two paths provided by the PCC (that joins the end nodes of a chord link) is longer in the DFS method. For example, compare the shortest path that joins the end-nodes of the dashed edge of Figure 3.19a with that of Figure 3.19b.

2) The BFS approach generates PCCs that usually covers edges in the same area while the DFS covers edges in different domains.

In terms of complexity, both approaches have the same complexity since each node is inserted in the queue once and the same number of comparisons is made. The complexity of the BFS method has been discussed in Section 3.2.3.

3.3.2 Discussion

The problem of finding the longest cycle in a planar graph is an NP-complete problem. The longest cycle in the graph G is called the circumference of the graph, denoted by circ(G). A Hamiltonian cycle is a cycle with size equal to the number of nodes. Our MCG rules in Section 3.1 give sufficient conditions to find the long cycles in the graph. However, they do not provide information on the size of the cycle to be generated. Many theorems have been introduced to find a good lower bound for the size of long cycles, e.g., [ChNi89]. MCG could be used in conjunction with these theorems, and our results can be updated until the lower bound in the theorem is reached.

The DFS approach appears to be suitable to produce long cycles, but unfortunately it is not guaranteed to find the longest cycle in a planar graph in a polynomial running time because the algorithm could be blocked (when other faces cannot be added to the current cycle). Using an exchange-and-repeat method based on the idea of backtracking (to get around the blocking point), it may be possible to find the longest cycles but will cause an exponential running time. The main problem that would affect finding the longest cycle is the number of shared edges between the faces. When the number of shared edges is more than one, then either of the two faces could be inside the cycle. In such a case, the algorithm should route around one of the two faces to find a longer route.

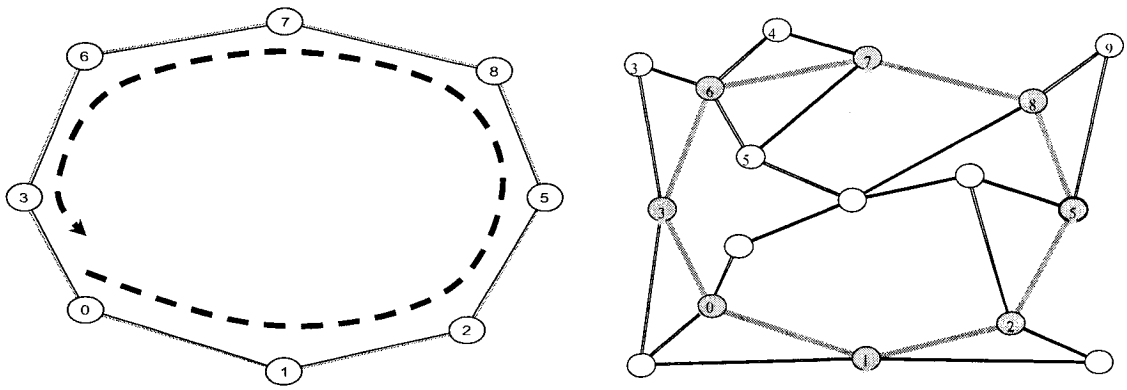
3.4 Concluding remarks

In this chapter, we have characterized the existence of PCCs using the graph faces in both the original G domain and the dual MC(G) domain. Although we are able to prove the mapping

properties from G to MC(G), a proof in the other direction is quite tedious due to the complexities to having to check all the different combinations and conditions of a given number of faces to form a PCC. So far, a complete proof has eluded us. We have also presented two formulations for finding PCCs using different searching techniques. Based on those formulations and by introducing different criteria for the size and for the next node choice we will have cycles with different properties for various applications. This will be discussed in both Chapters 4 and 5.

Chapter Four
Multicast Using PCCs

We would like to use virtual rings for multicasting as it has the advantage of surviving a single link failure. We want to investigate how PCCs studied in Chapter Three can be used to build a virtual ring for multicasting. We will evaluate its performance in terms of the complexity to obtain a "good" multicast cycle.



(a) (b)

Figure 4.1: a) A Multicast Ring,
(b) Embedding a Logical Multicast Ring in the Meshed Network

4.1 Multicasting using Cycles (MC)

In using a ring to multicast, a node having data to send to any other nodes in the network simply sends its data contained in a message in only one direction and waits for it to return. A returning message after one full rotation is a confirmation of the successful completion of multicasting. Figure 4.1a is an example of a multicast ring with 8 nodes. The multicasting ring can be protected against a single link/node failure by allowing the node to send the data in both directions of the ring in order to guarantee that each single node in the rest of the ring will receive the data. This is simply because there are two paths between each pair of nodes.

Figure 4.1b suggests that a procedure is required to embed a logical ring containing a given set of multicast nodes in a mesh network. This can be done by modifying the MCG algorithm

to seek out a given (but arbitrary) set of "on-cycle" nodes. In addition to the use of on-cycle links to provide a protected multicast, the chords (if any) can be used as redundant links to provide a shorter path in both the normal and failure operation.

There are some problems associated with finding MCs (Multicast Cycles). First, finding an MC is an NP-complete problem [Wils72], which is similar to the MST problem, and thus a good heuristic is required. Secondly, cycles have a longer transmission path length than trees on the average to reach all destinations. In other words, the mean time a message takes to reach all the multicast nodes in a tree is shorter. On the other hand, a cycle has an integrated protection mechanism while a tree needs an external protection mechanism.

4.2 Problem Statement

Based on the above discussion we can now formulate our problem as finding a cycle to contain a given subset of nodes in a general network graph. Formally, we have

Input:	The network topology with a set of N nodes and a set of M edges.
	A multicast set which is a subset of the network nodes.
Output:	A cycle path specification connecting all multicast nodes.
Constraints:	if possible, a cycle of no more than x hops.

We want to obtain a fast and efficient algorithm to determine the existence of a MC connection.

4.3 Implementation Approach

To contain all the multicast nodes, we can run the PCMCG algorithm using either one of the formulations (Alg_PCMCG or Alg_PCMCG2) presented in Section 3.2. To do this, we start with a face to generate one PCC at a time, and stop when the current cycle found includes all the multicast node members.

Since our PCMCG algorithm is very fast, we can repeat the above process with yet another face until all faces have been tried as a starting face. Therefore, more than one cycle may exist to contain the set of multicast nodes. A criteria can be employed during the searching technique to produce cycles of given properties (or to achieve a given objective). A typical objective in multicasting is to minimize the length of the cycle (in terms of the number (x) of hops).

Several modifications are required for the algorithm and its data structure. First we need a data structure (e.g. an array) to store the set of multicast nodes for comparison in the algorithm. Then, in order to check for the existence and validity of multicast node members, each node in the dual domain would need an extra parameter which is the set of nodes bounding each face. During each step of the algorithm, whenever a new node is added to the subgraph that represents the PCC in the dual domain, a comparison is made to check against

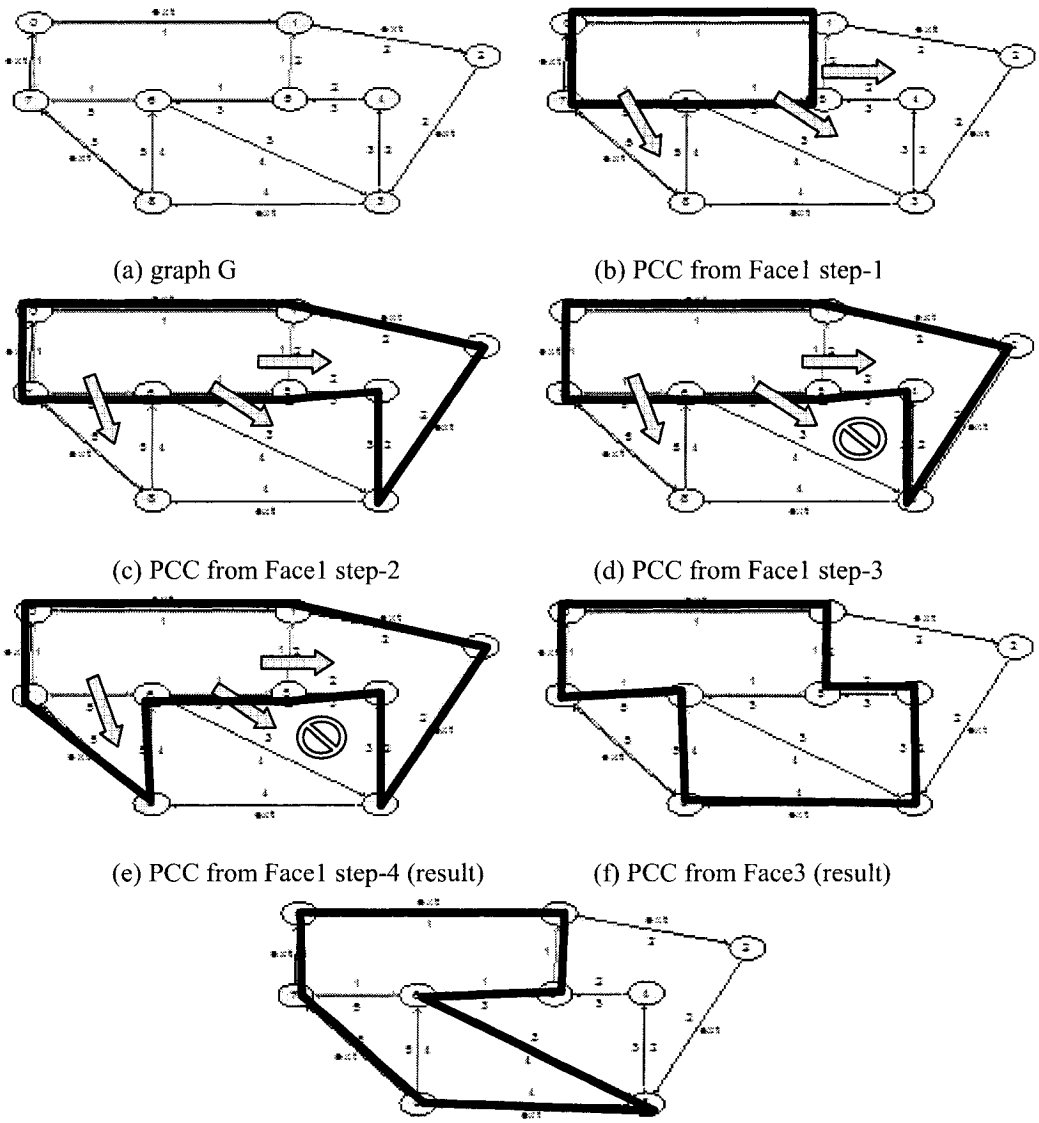
```
Alg_PMPCC
For all v in MC(G)Do
      Initialize array status[] of size equal to the number of nodes in MC(G) to not-visited.
      valid= Call Alg_PCMCG3(v,S) //calls Alg_PCMCG3 to mark the nodes that can form a PCC
                                  //with v as visited, S is the multicast members
      If (valid is true)
        PCC[i++]=find_cycle(status) // generates the PCC based on the resulted status value
Return PCC list
End Alg_PMPCC

Alg_PCMCG3 (v,S)
Begin Alg_PCMCG3
        S1=S
        Status[v]=visited
        Check_Delete(v,S1) // removes from S the nodes that are included in Face v
        Q.Push(v)
        while Q OR S1 is not empty do
                v=Q.pop
                get A'(v) and organize list of nodes (y) based on some ordering
                for all nodes y ∈ A'(v)
                        if node y is not visited then
                                if (cycle_regular(v,y,z) and status[z]=visited for some node z)
                                OR (node v and node y connected by multiple edges) then
                                status[y]= blocked,
                        else If (cycle_dashed(x,y,z) and labels(y) ∈ cycle labels)
                             OR (no edge between node v and node y)
                             status[y]= visited
                                Check_Delete(v,S1)
                                Q.push(y)
                        For all nodes y ∈ A'(v)
                                If status[y]= blocked and c(y) ≤1 then status[y]= covered

    if (S1 is empty)    return true
    else                return false
 End Alg_PCMCG3
```

Figure 4.2: The PMPCC Algorithm

the multicast nodes stored in the data structure that have not been checked out. The process stops whenever the entire multicast node set has been checked out.



(a) graph G

(b) PCC from Face1 step-1

(c) PCC from Face1 step-2

(d) PCC from Face1 step-3

(e) PCC from Face1 step-4 (result)

(f) PCC from Face3 (result)

(g) PCC from Face5 (result)

Figure 4.3: Example of Using PMPCC to Construct a Multicast Cycle

4.4 The Algorithm

Figure 4.2 presents the pseudo code for our modified algorithm as discussed in Section 4.3. Called the PMPCC (Protected Multicast using PCC) Algorithm, it uses the BFS method and starts with a set containing all the multicast nodes. As discussed before, whenever a member of the multicast node set is included in the PCC it is removed (checked out) from the set. The algorithm stops if either the set is empty or no further expansion is possible. The main

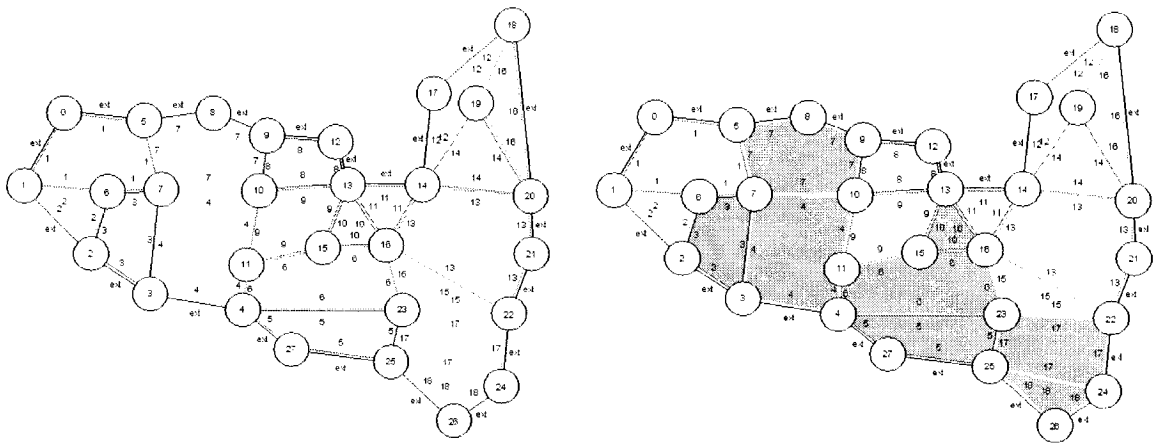
differences between this algorithm and the PCC finding algorithms in Chapter Three are the few lines shown in bold face.

4.5 Examples

Three examples are provided here. One is to illustrate the algorithm, the second is to consider a more realistic network, and the third example will be used in the performance evaluation section that follows.

Example 4.1: Simple Network

Figure 4.3a shows a set of multicast nodes (highlighted). Figure 4.3b-e provides a step-by-step illustration of how a PCC is generated to contain the set of multicast nodes from Face 1 when the PMPCC algorithm is run to provide the multicast cycle. Figure 4.3f and Figure 4.3g show the multicast cycle obtained when starting from face 3 and 5, respectively. The first result is a possible multicast cycle since it does include all the multicast nodes. So does Figure 4.3e. It is possible that a PCC cannot be found to include all the multicast nodes as in the case of Figure 4.3g. Here Figure 4.3e has 9-hop length cycle while and Figure 4.3f has 8-hop length cycle.



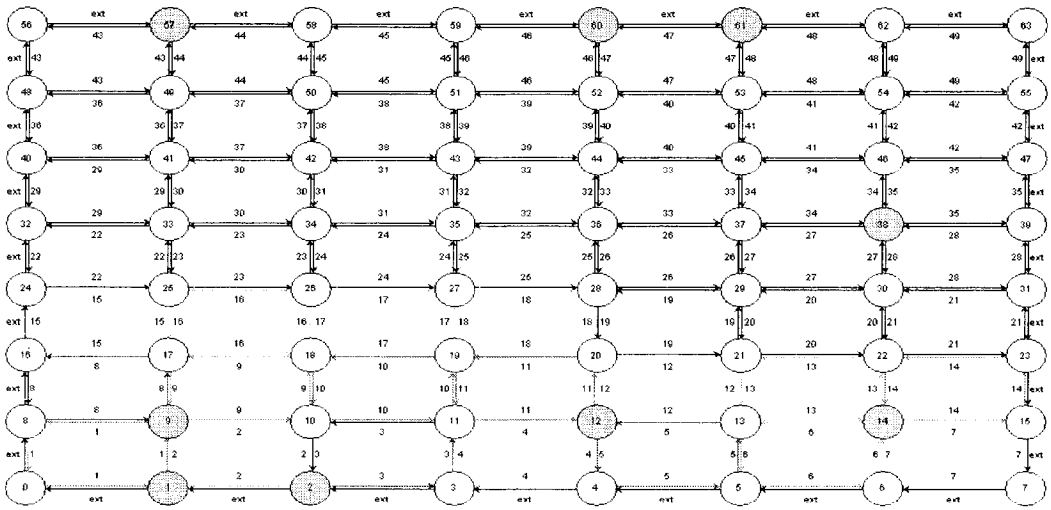
(a) Multicast nodes (b) Multicast cycle

Figure 4.4: Multicast Example in USA Long Haul Network

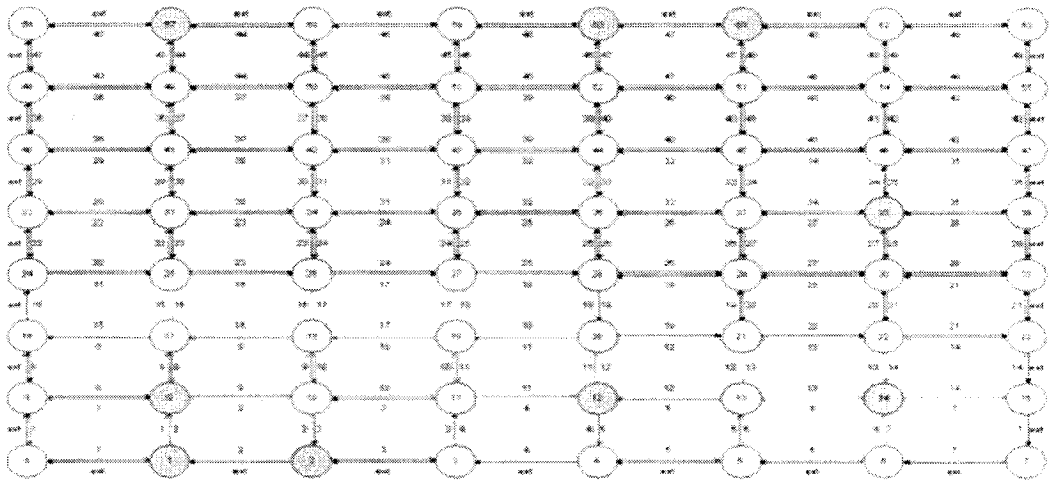
Example 4.2: USA Network

Figure 4.4a stipulates that the multicast node set {8, 9, 11, 16, 23, 24, 26} is required within the USA Network. Figure 4.4b illustrates the best multicast cycle generated using our algorithm. By examining and comparing this cycle with other cycles resulting from excusive enumeration we find this cycle is 3 hops longer than the optimal one. However, a post

processing for the PCC is possible to improve its size. This will be discussed in a later section in this chapter.



(a) Multicast nodes



(b) Multicast cycle

Figure 4.5: Multicast Example in a Grid Network with 64 Nodes (a) Multicast Nodes (b) Multicast Cycle

Example 4.3: The Grid Network

Figure 4.5a is a Grid network with N=64 nodes and the multicast set {1, 2, 9, 57, 61, 38, 60, 12, 14}. This is a large network which will allow us to have some estimate of the efficiency of the algorithm. Figure 4.5b is the best multicast set obtained. Again a quick and easy post processing is possible to optimize the size of the cycle by pruning the unused regions of the cycle like faces 22, 23, 24 which will be explained later in this chapter.

4.6 Performance Analysis

Since optimally embedding a multicast set within a mesh network can be an NP-complete problem, we evaluate the run time of our algorithm. We first use the results of Examples 4.2 and 4.3 as an introduction, then we provide a more systematic evaluation using random graphs later.

```
OUTPUT:
Please input the multicast node set ("," between node numbers, no space ):
8,9,11,16,23,24,26
Multicast PCC from node 5:3 4 5 6 7 10 17 18
Multicast PCC from node 12:5 7 8 9 10 11 12 13 14 15 17 18
Multicast PCC from node 13:5 7 8 9 10 11 12 13 14 15 17 18
Multicast PCC from node 14:5 7 8 9 10 11 12 13 14 15 17 18
Multicast PCC from node 15:3 4 6 7 10 13 14 15 17 18
Multicast PCC from node 17:5 7 8 9 10 11 12 13 14 15 17 18
Multicast PCC from node 18:5 7 8 9 10 11 12 13 14 15 17 18
Best multicast PCC:3 4 5 6 7 10 17 18
20 ms to find Multicast set!!
```

(a) Results for USA network

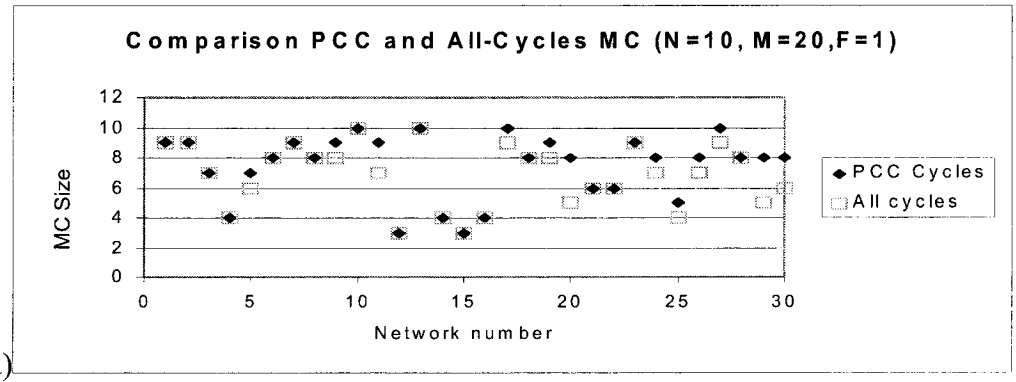
```
OUTPUT:
Please input the multicast node set ("," between node numbers, no space ):
1,2,9,57,61,38,60,12,14
Multicast PCC from node 1:1 2 3 4 5 6 7 8 10 12 14 15 17 19 21 22 24 26 28 2 9 31 33 35
36 38 40 43 45 47
Multicast PCC from node 2:1 2 3 4 5 6 7 9 11 13 15 16 18 20 21 23 25 27 29 3 0 32 34 35
37 39 41 43 44 46 48
Multicast PCC from node 3:1 2 3 4 5 6 7 8 10 12 14 15 17 19 21 22 24 26 28 2 9 31 33 35
36 38 40 43 45 47
...
Best multicast PCC:2 4 9 10 11 12 13 18 22 23 24 25 26 27 28 32 36 37 38 39 40 41 42 44
46 48
170 ms to find Multicast set!!
```

(b) Results for the Grid network

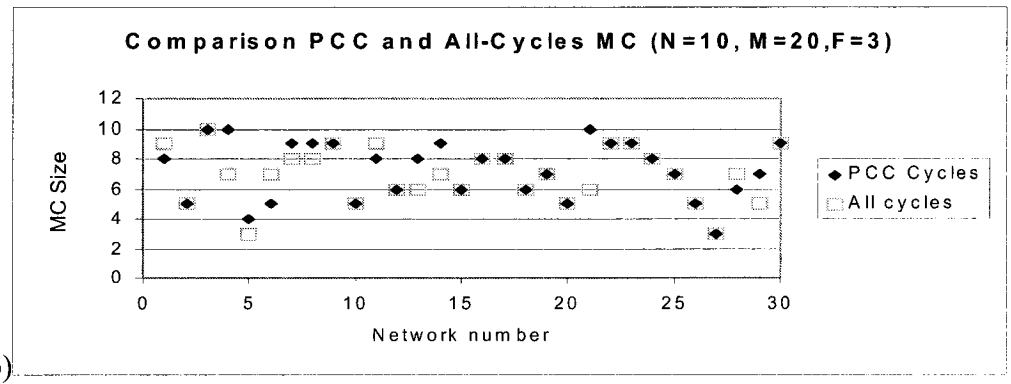
Figure 4.6: Performance of PMPCC on the USA and the Grid Network Examples

4.6.1 Time Complexity of Examples 4.2 and 4.3

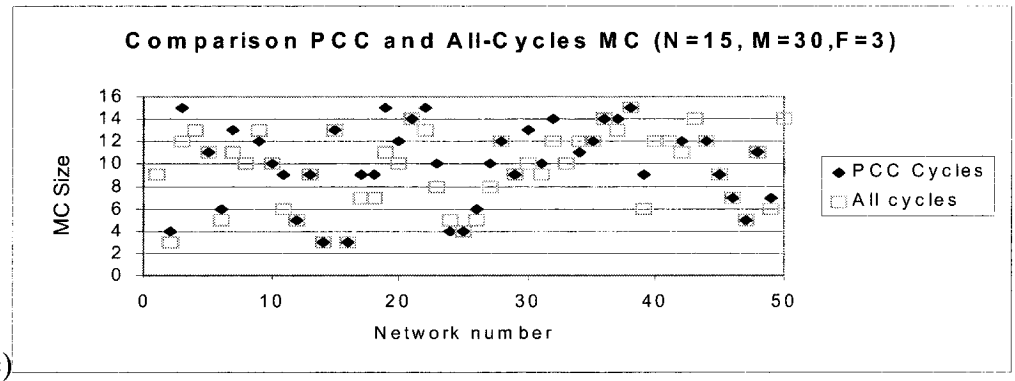
After running the PMPCC algorithm, we obtain a list of possible multicast cycles and the best (smallest) multicast cycle as shown in Figure 4.6. One can see the algorithm completes in a mere 20 ms for the real network example (USA).



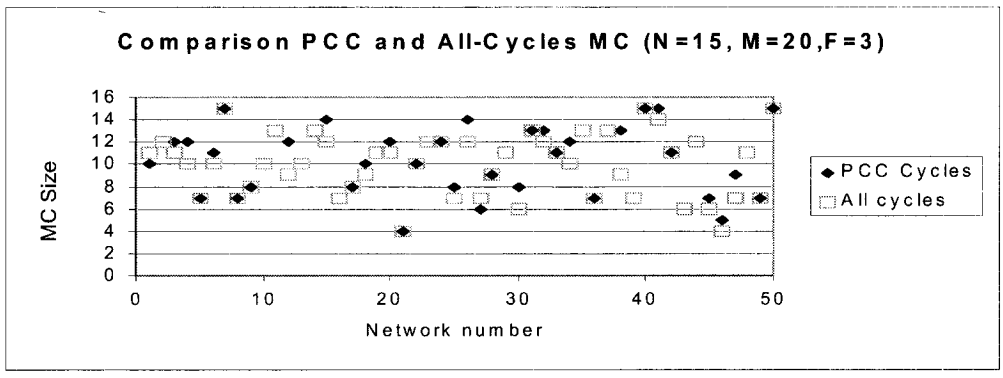
(a)



(b)



(c)



(d)

Figure 4.7: Comparison of the Best Generated Multicast Cycles Between All Cycles and PCC from Different Random Graphs

In Figure 4.6b we consider the algorithm performance on the larger Grid network with N=64 nodes, as described in Example 4.3 of the last section. In [HeFu99], a lower bound for

the number of cycles in a grid network is 1.41464^n=4,253,654,438 cycles. Thus, if we use an algorithm that tries to test all the cycles of the graph to find a multicast cycle would search for around 4 billion cycles, however using PMPCC searches up to 64*64= 4096! An illustration for the best-multicast cycle generated by the algorithm is shown in Figure 4.5b, and the time is merely 170ms, suggesting a polynomial time complexity for our PMPCC algorithm. The time complexity for the PMPCC algorithm is the same to that of finding a PCC which have been discussed section 3.2.3 and 3.2.4, i.e., it is of $O(M^2-MN)$.

4.6.2 Results for Random Graphs

We generate random graphs in order to compare our algorithm performance to the optimal performance (obtained by checking all cycle). The libraries in [LEDA06] are used to generate random planar 2-connected graphs. Then a random multicast set is generated from the network nodes.

We compare the best cycle generated by PMPCC with an optimal one obtained from exhaustively checking all possible cycles in the same random graph network. A cycle is considered optimal if it has a smallest number of hops.

In Figure 4.7 we compare the MC size (in terms of number of hops) obtained from All-Cycles (AC) and PMPCC for different network sizes. Let F denote the ratio between the maximum link capacity and the minimum link capacity over all the network links. The F parameter can be used as a criterion in choosing the next neighbor when there are multiple neighbors, thus affecting the searching process for PCCs. This will be considered in more details in the next chapter.

In each graph the horizontal axis gives the network number. All points in a given graph represent networks of the same size (with the same number of nodes, N, and the same number of links, M) (and different topologies) and with different randomly generated multicast set members. Figure 4.7a shows an example for N=10 nodes and M=20 edges with 30 networks generated. We can see in this figure that PMPCC successfully found an MC for each case where AC also found an MC. The sizes of the cycles are quite similar for many of the networks except for a few where the size of the PMPCC is higher than AC.

As mentioned in the algorithm discussion before PMPCC does not always succeed in finding an MC. This can be seen in Figure 4.7c for the first network where we have a square

and no diamond, which indicates that an MC has been found by AC but not by PMPCC.

Table 4-1: Comparison between the Average MC Hop Length for the Networks in Figure 4.7

N	M	F	MC average size	
			AC	PCC
10	20	1	6.92	7.53
10	20	3	7.06	7.46
15	20	3	9.50	9.94
15	30	3	9.96	10.38

Table 4-1 compares the AC and PMPCC algorithms further in terms of the MC average hop length for the networks considered in Figure 4.7. For N=10 and M=20, the average size of the optimal multicast cycle (found by AC) is around 7 hops long regardless of the capacity distribution (1 or 3). However, the average cycle size for the PMPCC algorithm for the same networks and same multicast sets was around 7.5. By examining the other two cases for N=15, we can see that the difference between the average lengths of the MC found by PMPCC and AC is about 5% of the AC size.

Although the comparison does indicate that PMPCC performance is a bit worse than AC, it can actually be improved to have the same or very similar performance. This will be discussed in Section 4.7.

4.6.3 Complexity Performance

We provide the following analysis to summarize our observations above. We know that for a planar graph with N nodes and f faces we have $f \leq 2N$-4. Thus, starting from a specific face and performing a searching technique, we will have at each step a possibly new PCC (thus a max of f PCCs). If we run the algorithm from each face in the graph, then the max number of steps is $O(N^2)$, since $f^2 \leq (2N\text{-}4)^2$. Thus, the number of PCCs generated is bounded by $(2N\text{-}4)^2$ when using PMPCC.

The previous two sub-sections provide some comparison examples between the theoretical

number of PCC with the total number of cycles in planar graphs. It has been shown [HeFu99] that the max number of simple cycles in a planar graphs with N nodes (denoted by C(N)) for a general planar graph is $2.28^N \leq C(\mathbf{N}) \leq 3.37^N$, and for a grid graphs is $1.414^N \leq C(N) \leq 1.935^N$. Thus, if a network has N=28 nodes, then we would have the max number of cycles in a general planar graph within the range $10523875848 < C(N) < 593796000000000$, while PMPCC would generate a maximum of $(2*28-4)^2$ =2704 PCCs.

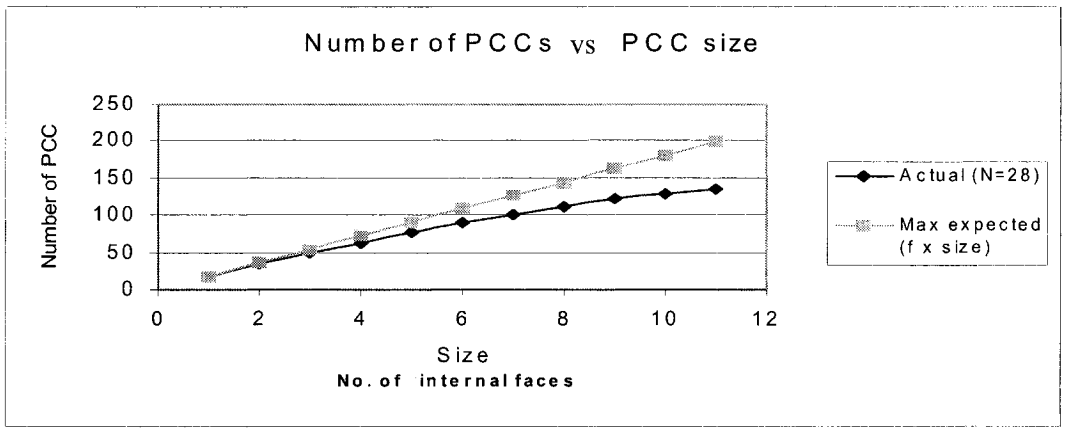


Figure 4.8: Actual and Upper Bound for the Number of PCCs Generated by PMPCC

In Figure 4.8, we compare further the actual number of cycles generated by PCMCG with this upper bound for a graph with 28 nodes. The upper bound is the theoretical maximum expected for the number of PCCs which is the total number of faces times the number of internal faces, i.e. f x size. On the x-axis, we show the number of internal faces in the PCC and in the y-axis we show the number of PCCs. The two curves show the maximum theoretical number of PCC and the actual number obtained from running the algorithm.

The complexity is also reflected in the run time of our algorithm. For networks with less than 30 nodes and nodal degree around 3, the time to build the modified cycle graph is around 10 milliseconds. Referring to Figure 4.8, the average time to compute all PCCs starting from a specific face is usually less than 15 milliseconds. Then the time to compute all PCCs is less than 15 faces * 15 ms/face = 225 milliseconds in comparison with a very long time (on the order of minutes) to compute all cycles.

We now discuss the complexity for improving the size of the PCCs which will be

discussed in the following section. In brief, in both algorithms we require visiting all the face that are internal faces that are marked as visited (max is $f=n'$). For each such face we traverse each adjacent face for it (average is D'_{avg} in MC(G)) and each edge in that face. Thus the complexity for reducing the number of faces is $n'.D'^{2}_{avg}= n'.(2m'/n')^2=2m'^2/n'$ (where n' and m' are the number of nodes and edges in the MC(G) graph). Regarding the other algorithm to allow the cycle of faces, we have similar parameters and thus similar complexity.

4.7 Optimizing PCC for Multicasting

Previous comparison results suggest possible improvements to the PCCs to make them more efficient for use in multicasting. This is discussed in the following.

4.7.1 Reducing the Number of Faces

Since our algorithms so far (both PMPCC and the previous version in Ch.3) are formulated to explore the whole network graph, including areas that do not have multicast nodes, the resulted multicast cycle could contain extra faces whose removal would improve the size of the cycle (i.e. shorter circumference). For example, in Figure 4.4b we could consider the multicast cycle without Face*3* which in turn will reduce the cycle length by two edges.

```
Alg_Face-Optimize1
For each node v in MC(G) such that
  It belongs to the solution set AND is adjacent to only one member in
  the solution set, do
      For all the edges of the corresponding face to v in G
        • If none of the nodes of those edges belongs to the
          multicast set, then remove it from the solution cycle
```

Figure 4.9: Optimize the Number of Faces Algorithm (Face-Optimize1)

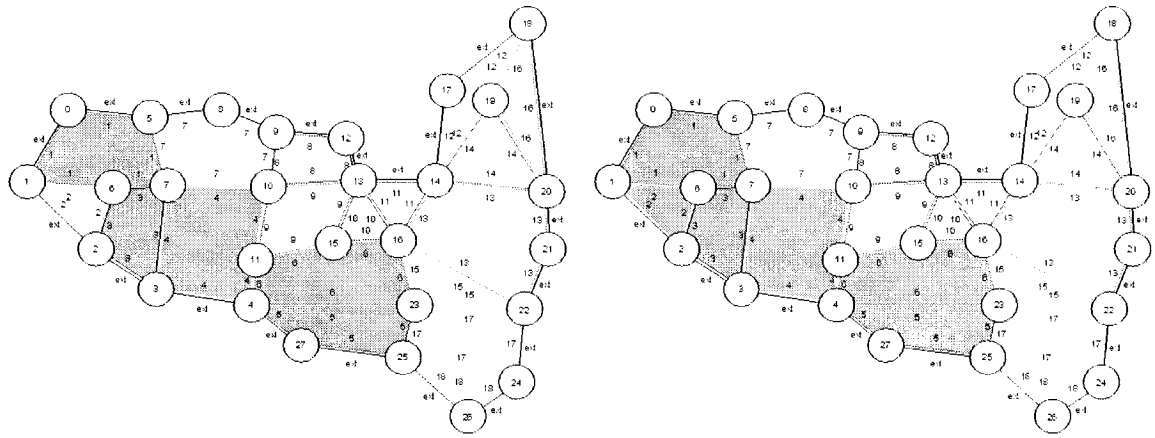
Figure 4.9 shows a simple algorithm that examines all the visited nodes of degree 1 in the solution set subgraph and checks if the corresponding face has any member of the multicast set nodes. If not, it removes the face from the multicast cycle. Otherwise, it keeps it. This algorithm should be repeated until no more faces can be removed.

4.7.2 Allowing Cycles of Faces

By definition, a PCC is not supposed to include a cycle of faces. This might affect the

efficiency of the cycle (in terms of its size) when used for multicasting, as there might be a shorter path when a cycle of faces is allowed. For example, if we include Face 2 in the cycle in Figure 4.10, then a shorter path from node 1 to node 2 can be found when using the direct edge rather than using two edges.

Including a face outside the cycle could reduce the cycle size if it is connected to at least two internal faces and if the number of unshared edges is less than the number of shared edges. This suggests the face-processing can be modified as follows. For each internal face, check the external neighbor faces and test if their inclusion would reduce the size of the cycle.



(a) (b)

Figure 4.10: Allowing a Cycle of Faces to Reduce the Cycle Size

```
Alg_Face-Optimize2

step 1
  o    For each node v in MC(G) that belongs to the solution set, do
           ▪ look at the neighbors that are not in the solution set
             and adjacent to v and include them in a frequency list
step 2
  o    consider all the neighbor faces that have frequency > 1
  o    for each such face subtract the number of edges shared with the
       solution set from the number of non-shared edges
       ▪ note: shared edges shouldn't have a multicast node in common
  o    if the difference is 1, include that face in the solution set
```

Figure 4.11: Allow Cycle of Faces Algorithm (Face-Optimize2)

Figure 4.11 is the pseudo-code for the algorithm to search among cycles of faces whose

center node is not in the multicast set of nodes. The algorithm should be repeated until no more faces can be added

4.8 Dynamic Membership

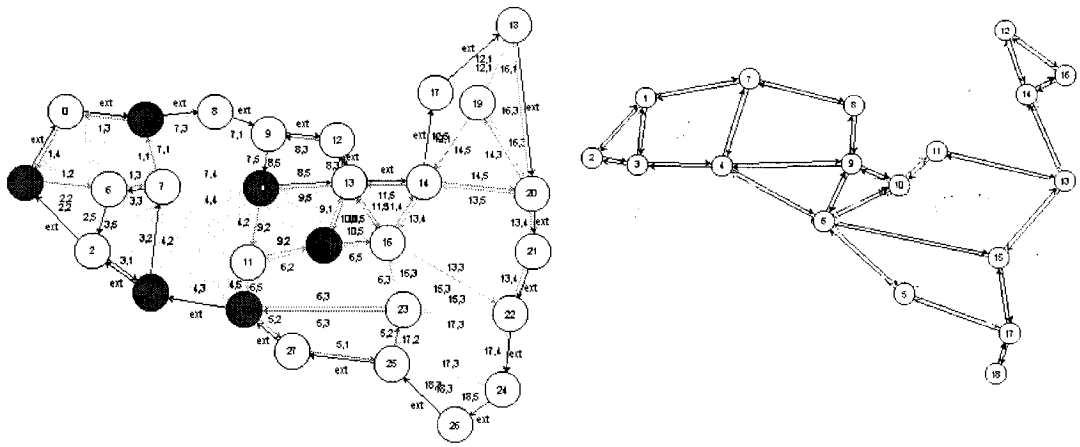
Since members can join and leave a multicast set, we need to consider operations of adding or dropping a node and how the cycle can be updated. Other related issues involve the adding or severing of a link, and changes in link cost that might affect the cycle.

The most simple-minded approach to the above problem is to treat the network as a new problem after each change, and therefore just rerun the algorithm after the changes to obtain a new MC configuration. The problem in doing this is that many connections may need to be re-routed and the packets already in the pipe retransmitted. Therefore, we shall provide in the following a simple make-and-break operation to make the transition to the new MC configuration after a new node joins the multicast set. make-and-break simply refer to making the new solution before breaking the old one when a change occurs to the current state of the current solution.

```
For each node v in the new multicast set but not in the old set
   Identify all the neighbor faces to v in G (call this set NF_v)
   For each face f in NF_v
        -Find the shortest path from the node corresponding to f in
        the weighted MC(G) to the current nodes corresponding to the
        current solution
    Choose the shortest among this set and add all the corresponding
    faces to the multicast cycle faces
```

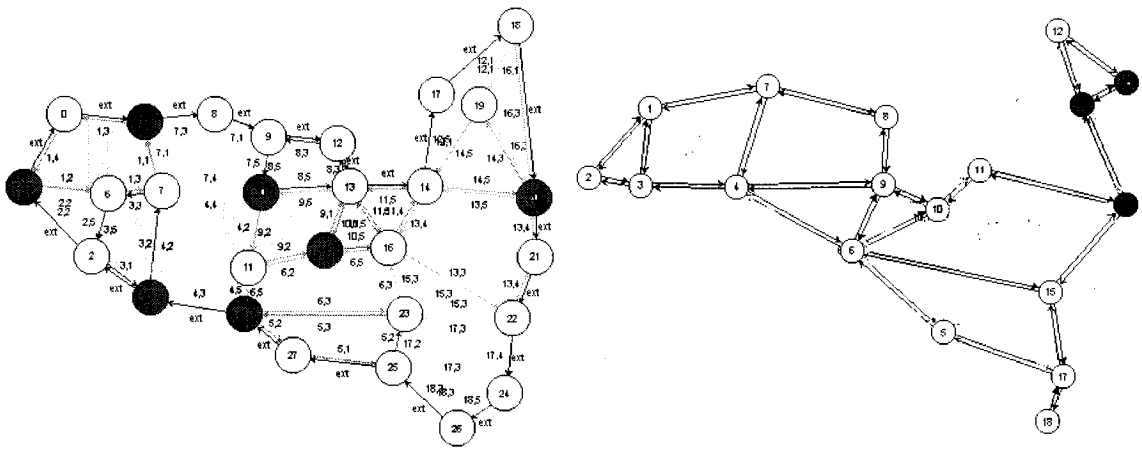
Figure 4.12: Updating the Multicast Cycle when New Multicast Nodes Join

Figure 4.12 is the pseudo code to illustrate the operations which lead to the new multicast cycle. In essence, the shortest path is first found in the "weighted" MC(G) graph from each neighbor face (node in MC(G)) of the new multicast node to the current subgraph (representing the current multicast cycle). Then the shortest of these paths is used. A weighted graph in this discussion is one where all edges are weighted with respect to the number of shared and non-shared edges between the faces.



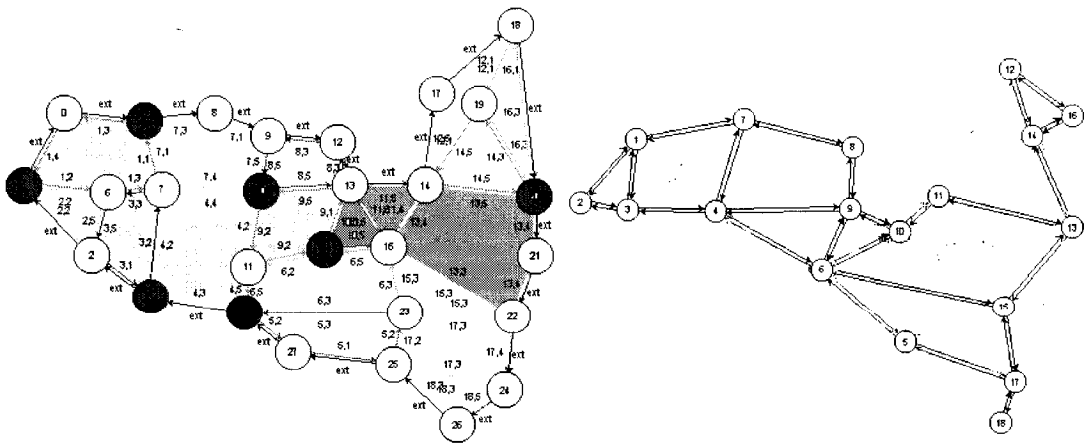
(a) G (b) MC(G)

Figure 4.13: Multicast Cycle for a Set of Multicast Nodes in G and MC(G)



(a) G (b) MC(G)

Figure 4.14: New Node Wants to Join the Multicast Set



(a) G (b) MC(G)

Figure 4.15: Multicast Cycle for a Set of Multicast Nodes in G and MCG

Figure 4.13 to Figure 4.15 are used to illustrate the steps in the algorithm. Assume we have the current solution for a given multicast set {1, 3, 4, 5, 10, 15} shown in yellow (Figure 4.13a), and the corresponding MC(G) in Figure 4.13b where the green nodes are the internal faces in the multicast cycle. When node 20 wants to join this multicast set (Figure 4.14), the algorithm identifies all the neighbor faces next to 20 {in this case, they are faces 13, 14 and 16). Then, we find the shortest path from each of the three nodes to the green nodes. Then add the nodes along that path (10, 11, 13) to the current solution (Figure 4.15) and the new MC is now defined.

The weights for each regular edge $\{u, v\}$ on the MC(G) is calculated by weight(u, v) = e(u)+e(v) - number of edges between node u and v where e(n) is the number of edges face n has in G.

4.9 Concluding Remarks

We have presented algorithms to find multicasting cycles, which are protected against single failure. The use of chords is possible as we have mentioned in the beginning of the chapter to have further improvement in the protection capability provided by the cycle. We have shown how different simple post-processing algorithms can be used to improve efficiency of the produced PCCs. Further improvements are possible but would increase the complexity of the solution. In summary, our multicasting technique has the properties of tree-like (fast) construction and ring-like (reliable) protection.

Chapter Five
Planar Graph Protection using PCCs

In this chapter, we shall study the application of PCCs for network protection, and hence the capability of an MCG-based algorithm to generate a special set of p-cycles. This has the advantage of bypassing the task of having to generate all p-cycles in the network, which has an exponentially increasing complexity with the size of the network (and is thus an NP-Complete problem in general). Our algorithm here has a polynomial complexity and hence is efficient and scalable in providing cycle-based network protection especially in optical network applications (i.e., with a response time on the order 50 ms).

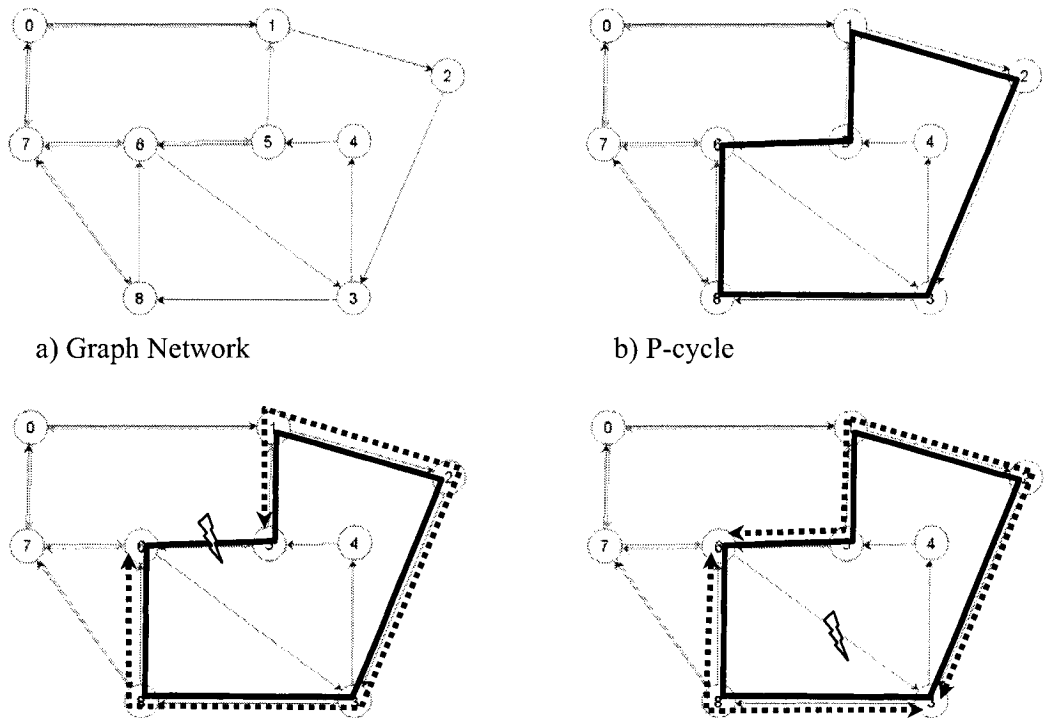
5.1 Cycle-Based Protection

In the cycle-based network protection methods we want to find an optimum (with respect to a performance measure) set of cycles to cover (protect) all network links. There are two types of protection (coverage). In ring protection, the link is just a part of the cycle (i.e., an on-cycle link) and it is protected by the path that traverses the cycle in the other direction. In p-cycle protection, however, a link can be either an on-cycle link or a chord (straddling link), i.e a link that is directly connecting two nodes on the cycle. Figure 5.1 illustrates the scenarios of failures of an on-cycle link and a chord.

For example a logical p-cycle as in Figure 5.1b can be set up in the mesh optical network where all links are bi-directional. When link 5-6 fails, the nodes would detect the absence of the signal (and therefore a failure). By consulting the database at node 6, the data in the direction of 6-5 can be routed counterclockwise along the path of 6-8-3-2-1-5 (Figure 5.1c) using pre-established wavelength, and similarly the traffic going in the direction of 5-6 can be routed clockwise along the path 5-1-2-3-8-6. In the case where the failed link is a chord we have two choices when consulting the database at the failed nodes of that link (see Figure 5.1d).

To generate an optimum set of p-cycles using spare bandwidth as a performance measure, reducing the number of protection cycles usually reduces the spare bandwidth requirement and always makes it easier to manage and control the survivability of the network. In fact, if

a Hamiltonian network (as defined in Chapter Two) is homogeneous (i.e., all the links have the same working capacity), then the optimum solution is a single Hamiltonian cycle which protects all the links of the network with the minimum amount of spare capacity used [HuCo01]. An example is the cycle 0-1-2-3-4-5-6-8-7-0 in Figure 5.1. Assuming all the edges in the network have 1 unit of working capacity, this cycle is optimum in the sense that it requires the least spare capacity required to protect the whole network.



a) Graph Network

b) P-cycle

c) On-cycle link failure recovery (one path exists)

d) Chord link failure recovery (two paths exists)

Figure 5.1: P-Cycle Operation Example

However, for large networks this solution (if it exists) is impractical as discussed in Chapter One (Section 1.5). This is because such a solution will incur a long propagation delay in the recovery of one cycle link by having to traverse all the nodes and links in the opposite direction in the p-cycle (to reach the other end of the failed link). Another disadvantage for long p-cycles is that they reduce the probability of the network being able to survive a dual failure.

The best set of p-cycles that protects the network can be found from a candidate set of

cycles through optimization. The "biggest" candidate set is the set of all cycles in the graph network, which obviously contains the most optimal set of cycles to protect the whole network. Unfortunately, the size of this set grows exponentially with the network size, and so does the complexity of the search algorithm. So it is natural to address the following issues related to our PCCs:

a) How good are the PCCs as a candidate set when compared with the set of all cycles?
b) What different searching techniques, when employed in the generation of PCCs, would improve the results?
c) How much do restrictions on the size of the PCC (or regular cycle) affect the optimal solution?
d) How far are all the cycle based methods (including our PCC methods) from the global optimal solution obtained from all paths enumerations?

5.2 Problem Statement

Based on the issues we want to address, we formulate our problem as finding a set of cycles (candidate set) that can be used to find an optimal solution for the network protection but with various constraints built in.

Input: A network topology of N nodes and M edges, along with the cost and the working capacity of each edge.

Output: A P-cycle candidate set.

Constraints: These can appear in various forms that would affect the type of cycles to be obtained. For example, cycles with a maximum number of hops, or a criteria that would exclude certain next neighbor nodes to be searched.

5.3 Implementation Approach

We first run the PMPCC algorithm (formulation 1 or 2 presented in Chapter Three) to find PCCs that can be used in the candidate set. Since allowing cycles with no size limitations may generate long cycles (eventually Hamiltonians) in general, we would like to set a limit on the size of cycles to be generated. However, we would also like to investigate the effect of using different size limits to generate PCCs to cover the network. A sample USA Long Haul Network is used in various case studies to demonstrate the effect, and randomly generated networks with a similar size and capacity distribution are used to demonstrate the general trend of the performance.

Since the formulations in Chapter Three do not use any specific criteria for choosing the next node, we shall refine the searching techniques here to be used in generating the PCCs.

We shall evaluate later the performance of these techniques in their capability to find the optimal solution in networks that are randomly created with various capacity distributions.

The Modeling Language for Mathematical Programming (AMPL) [AMPL06] will be used to find the optimal subset of cycles that best protects the network. This simple and structured modeling language uses the CPLEX optimization software package [CPLE06]. Our objective is to minimize the total spare capacity in the network links assuming enough capacity always exists in each link.

We then discuss the effect of variation in the network capacity distribution on these results. After that, we study how far the cycle-based methods are from the global optimal solution that results from enumerating all the network paths and solving the optimization problem to find the most optimal set.

5.4 Size-Restricted PCC and PCC Coverage

The basic algorithm in Chapter 3 is now modified to provide PCC coverage and to limit the size of the generated PCCs if so desired. This is done as follows.

5.4.1 Limiting the PCC Size

The idea is rather simple. A size parameter is added to the algorithm so that the PCC size can be checked after each node is marked as visited; the algorithm exits when the target size is reached. The pseudo code of this algorithm (called Alg_PCMCG4) is shown in Figure 5.2.

There can be variations such as limiting the number of levels used in the BFS/DFS search tree, the number of internal faces of a generated cycle, or the number of hops of a generated cycle. Since all these variations lead to a cycle with a limited size, we simply refer to them as the size parameter. We shall use them in different contexts to show the effect of limiting the size of the PCC.

5.4.2 PCC Coverage

Based on the idea of a cycle cover in which one finds a set of cycles such that each edge appears at least once in a cycle, either of the two formulations shown in Chapter Three can be used to generate a PCC cover. We run the algorithm until no further addition of faces is possible. The algorithm is then re-run on the remaining graph faces. The pseudo code for this is shown in Figure 5.3.

```
Initialize array status[] of size equal to the number of nodes in MC(G) to not-
visited.
Call Alg_PCMCG4(v,s)// calls Alg_PCMCG3 to mark the nodes that can form a
                    // PCC with v as visited. The cycle size is limited to
                     // an internal faces
PCC=find_cycle(status)//generates the PCC based on the resulted status value
```

```
Alg_PCMCG4 (v,s)
Begin Alg_PCMCG4
    Status[v]=visited
    Q.Push(v)
    current_size=1
    while Q is not empty and current_size<s do
       v=Q.pop
       get A'(v) and organize list of nodes (y) based on some ordering
       for all y ∈ A'(v)
           if y is not visited then
              if (cycle_regular(v,y,z) and status[z]=visited for some node z )
                  OR (node v and node y connected by multiple edges) then
                       status[y]= blocked,
              else If (cycle_dashed(x,y,z) and labels(y∈cycle labels)
                       OR (no edge between node v and node y)
                            status[y]= visited
                            Q.push(y)
              For all nodes y ∈ A'(v)
                       If status[y]= blocked and c(y) ≤1 then status[y]= covered
              current_size= current_size+1
End Alg_PCMCG4
```

Figure 5.2: Pseudo Code of PCMCG4 to Find PCC with Limited Size

```
Initialize array status[] of size equal to the number of nodes in MC(G) to not-visited.
While status contains not visited node
       V= not visited node
       Call Alg_PCMCG(v)       //calls Alg_PCMCG to mark the nodes that can form aPCC
                               // with v as visited
       PCC=find_cycle(status) // generates the PCC based on the resulting
                               // status values
       Block_Faces(Status)     // the faces that have been used in the generated PCC
                               // will be marked as blocked, so they will no be used
                               // in the new cycle
End while
```

Figure 5.3: Using PCMCG to Find PCC Cover

A combination of the above criteria could be used to find a PCC cover with a limit on the cycle size. Note that apart from some links, we do not use any faces inside the second cycle. The following examples explain this idea with a relatively large network graph example.

5.4.3 Criteria and Examples

There can be many different combinations of PCC coverage with different size limits. We shall use the following set of criteria on the extended USA Long Haul Network with 49 nodes and 66 edges shown in Figure 2.6. There are 5 criteria, each with fewer branches in each BFS level. The results are shown in Fig. 5.4 in which different cycles are presented in different shading levels. This network is the same as Fig. 2.6 which shows the face numbers much more clearly.

1) Using no criteria

This is the standard PCC cover where we have no restriction on the number of levels to be searched, and its results will be used as a reference. The idea is to search the whole network for the biggest PCC (recall we repeat the search by starting from each face so the biggest cycle can be selected). Then we repeat the above procedure on the remaining part of the network not covered by the previous PCC found.

Example: Figure 5.4a shows that Faces 19 and 21 cannot be covered by the biggest (first) PCC. Therefore, a second PCC is generated when we re-run the algorithm on these two faces. Consequently, we need 2 PCCs to cover the whole network.

2) Using 4-level BFS

The procedure is similar to (1) except that the BFS algorithm is not allowed to go more than 4 levels (i.e. its depth) of the search tree.

Example: Figure 5.4b shows that 5 connected parts (shown in different shading levels) remain to be covered after the biggest (first) PCC is obtained. Therefore, we have to run the algorithm for each face. Consequently, we need 6 PCCs to cover the whole network. Note that a connected part refer to region of faces next to each other. In our example here, each connected part happens to be the special case of a single face.

3) Using 3-level BFS

The procedure is the same as (2) but only 3 levels are used. Again, the algorithm is repeated recursively for each remaining parts until all connected parts are covered.

Example: Figure 5.4c shows that there are 4 connected parts (shown in different shading levels) remain to be covered after the biggest (first) PCC is obtained. Therefore, we have to

run the algorithm for each part. Consequently, we need 5 PCCs to cover the whole network. Notice that 3 connected parts are single faces here and 1 connected part is consists of 4 faces.

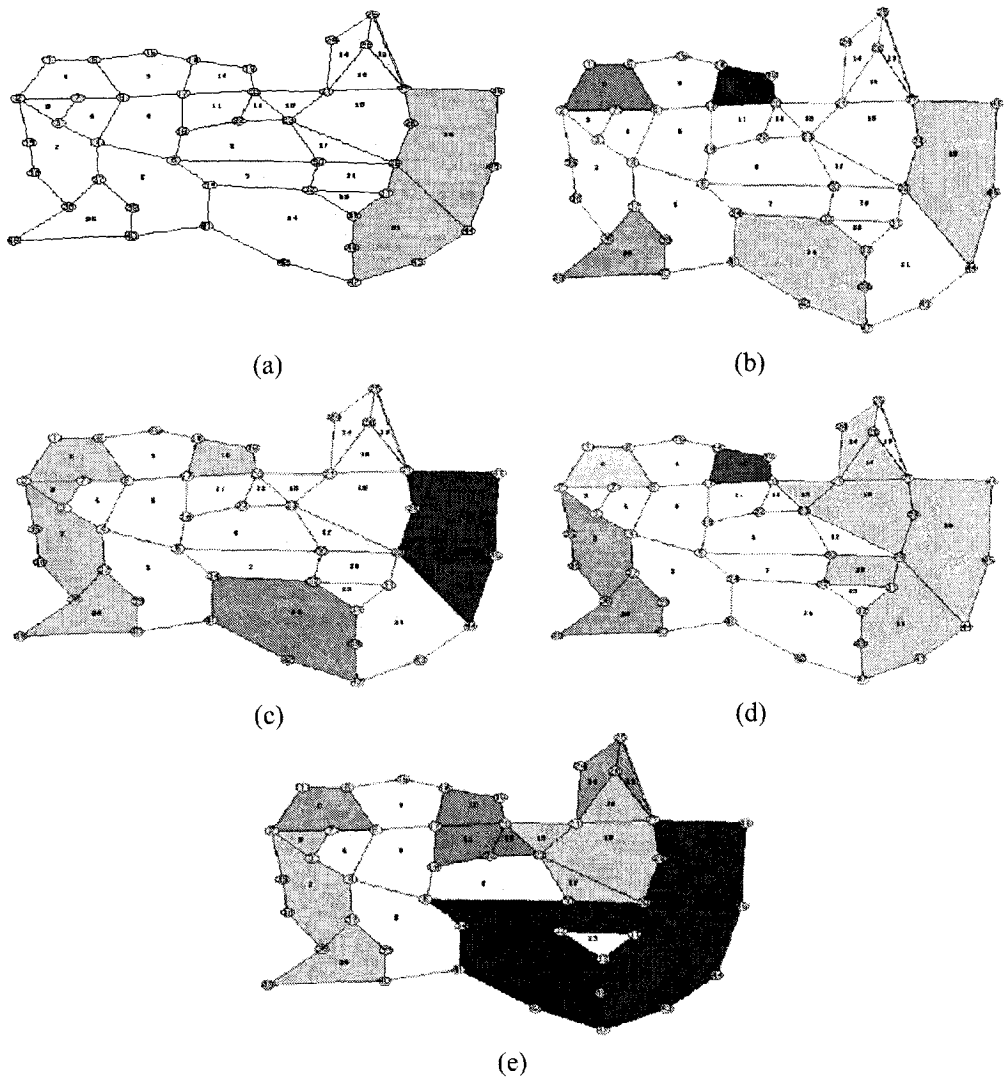


Figure 5.4: PCCs Generated using Different Criteria

4) Using 2-level BFS

The procedure is the same as (2) but only 2 levels are used. Again, the algorithm is repeated recursively for each remaining parts until all connected parts are covered.

Example: Figure 5.4d shows that there are 4 connected parts (shown in different shading levels) remain to be covered after the biggest (first) PCC is obtained. Therefore, we have to run the algorithm for each part. Consequently, we need 6 PCCs to cover the whole network. Notice than one of the connected parts (consists of 9 faces) needs two PCC to be protected.

5) Using 1-level BFS

The procedure is the same as (2) but only 1 levels are used. Again, the algorithm is repeated recursively for each remaining parts until all connected parts are covered.

Example: Figure 5.4e shows that there are 2 connected parts (shown in different shading levels) remain to be covered after the biggest (first) PCC is obtained. Therefore, we have to run the algorithm for each part. In this case, several connected part need to repeat the procedure recursively and therefore more cycles to cover. Consequently, we need 6 PCCs to cover the whole network.

In summary, when using a breadth-limited BFS in our algorithm, the more restrictive on the breadth, the more PCCs are required to cover the mesh network. Similarly, using a depth-limited DFS will also result in more PCCs (examples not shown here). Our MCG algorithm appears to be very flexible to provide a set of p-cycles according to different types of size limits On the other hand, our investigation above is limited to unit working and protection capacity if we are to provide 100% restorability to the network. If each link has a different working and spare capacity, not only 100%-restorability is not guaranteed, we are also faced an optimization problem to get the best to satisfy the required capacity. This will be presented in the following section.

5.5 PCC Size vs. Optimal Coverage

When each link has a different working and spare capacity from others in the network, the biggest PCC cannot be obtained easily as the first PCC because a PCC is created only when all on-cycle links must have the maximum of the link bandwidth. Finding the right combination is an NP-complete problem. To find out the optimum coverage, we could first generate an arbitrary set of PCC so long as they cover the network (i.e. each link appear at least one PCC. This can be done guaranteed by generating PCCs starting from each face at a time. This set of PCCs can then be optimized using optimization model to get the best subset that would have the least capacity.

With this procedure, we want to investigate the relation between the PCC cycle size (in hops) and the optimal solution that can be obtained from this cycle size limit. We shall investigate it under the assumptions of identical working capacities and uniformly distributed working capacities.

5.5.1 Networks with Identical Working Capacities and the Longest Cycle

We first start with the simpler case in which each link in the network has the same working capacity. Our objective is to find the largest cycle to protect the whole network in order to demonstrate the good PCMCG performance in finding long cycles. As explained before this objective is equivalent to finding the minimum spare capacity to protect the network when the Hamiltonian network has a unit working capacity on each of its links.

We have used the ALG- PCMCG3 algorithm on 1000 randomly generated planar networks graphs (as discussed in Section 2.5). In each graph, we compare the best solution (largest cycle) with the largest possible cycle obtained from excessive enumeration for all possible cycles. Our ALG-PCMCG3 was able to find the same solution in around 94% of these cases (i.e. around 940 cases)

5.5.2 Networks with Uniformly Distributed Working Capacities

For the networks where the working capacities are different, then we can use the ALG-PCMCG3 algorithm to generate a set of all possible PCC with all possible size value (in term of number of internal faces). The ILP (Integer Linear Programming) [Grov98] is then used to optimize this set as follows.

Let C be the set of candidate cycles, E the set of all network links, $M = |E|$ the number of network links, c_{sj} and c_{wj} the amount of spare and working capacity on link j respectively, c_j the cost of link j, and n_i the number of unit capacity copies of cycle i. Our objective is to optimize the amount of spare capacity as follows.

$$\text{Minimize} \qquad \sum_{j=1}^{M} c_j c_{sj} \qquad (5.1a)$$

$$\text{s.t.} \qquad s_j = \sum_{i=1}^{|P|} p_{i,j} \cdot n_i \qquad \forall (j \in E) \qquad (5.1b)$$

$$c_{wj} \le \sum_{i=1}^{|C|} x_{i,j} \cdot n_i \qquad \forall (j \in E) \qquad (5.1c)$$

$$n_i \ge 0 \qquad \forall i = 1,2,\cdots|C| \qquad (5.1d)$$

where the variable $p_{i,j}$ is 1 if cycle i passes over link j; otherwise it is 0; the variable $x_{i,j}$ is the number of paths that a single copy of p-cycle i provides for the restoration of link j in cycle i for all (i, j). This variable is 0 if one or both of the end nodes of the prospective failure link j are not on the cycle i, 1 if both end nodes are adjacent to each other on the cycle, and 2 if both of the failure link end-nodes are not adjacent to each other on the cycle (this is the case that link j is a straddling link of cycle i.). Both variables $x_{i,j}$ and $p_{i,j}$ are evaluated in advance for each cycle in C.

5.5.2.1 Performance Evaluation

We use the above optimization to evaluate the effect of PCC size on a simplified version of the USA Long Haul Network with N=28 nodes and M=45 links (Figure 2.7). The working capacity of each link is assigned an integer value randomly but uniformly between an upper and a lower bound as follows:

1) Identical unit capacities (no randomness; used as reference)
2) Integer randomly between 1 and 2
3) Integer randomly between 1 and 3
4) Integer randomly between 1 and 4
5) Integer randomly between 1 and 5

Figure 5.5 shows the optimal spare capacity as a function of the PCC size in terms of the maximum number of faces that can be included in each PCC. For the reference case of unity capacity (no randomness), the spare capacity is a decreasing function of the PCC size. One sees that 60 units of spare capacity are can be obtained from cycles with one face. This decreases to around 30 spare units for cycles with 11 faces. Similar observations are made when there are more working capacities allowed. The networks of case 5 achieve the highest optimal capacity (150 units for PCC size of 1 down to 75 units for PCC size of 11). Also it would appear that the amount of spare capacity gained in uniformly increasing as the randomness is increased.

Figure 5.6 shows the percentage improvement in the optimal spare capacity as a function of the cycle size transition. For example, there are around 18 units reduction in Case 1)) when we increase the PCC size from 1 to 2 (indicated by "1-2" label in the horizontal axis). This represents an 27% (16/60) improvement. As one can see, most reduction (around 10%-

27%) occurs when the PCC size is small (less than 4), but getting less (around 5%) when PCC size is increasing. Similar behavior can be seen for the other curves. This suggests that using small cycles would be enough to get a good suboptimal solution.

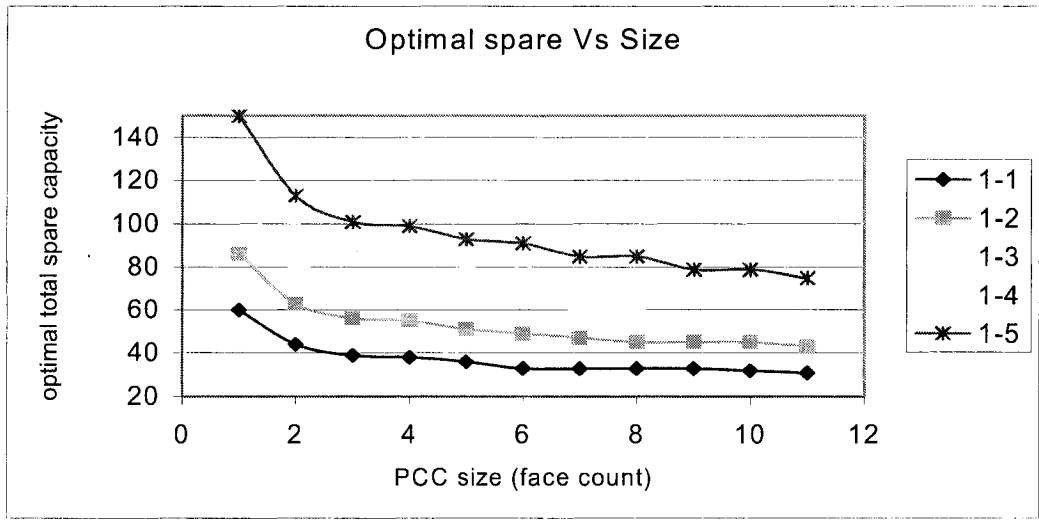


Figure 5.5: Optimal Spare Capacity with Respect to PCC Size (F=1, ... 5)

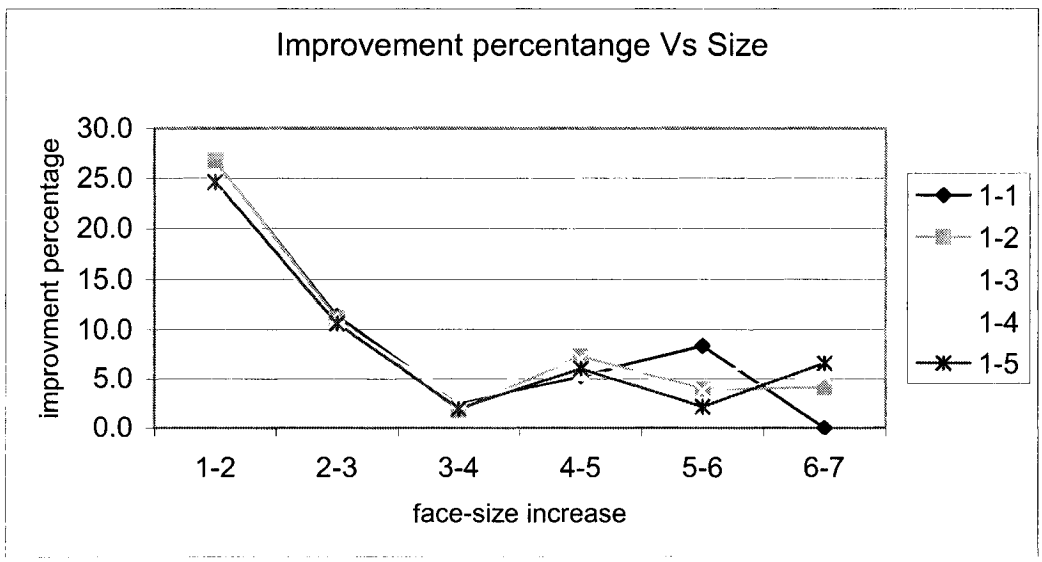


Figure 5.6: Improvement in Optimal Spare Capacity as a Percentage vs. the Increase in the Cycle Size in Terms of Faces

Figure 5.6 also shows the percentage improvement fluctuates at bigger PCC size as we are only investigating a particular network. To appreciate the general behavior better, we have

randomly generated 100 simple planar biconnected networks, each with 20 nodes and 29 edges. There are 4 working capacity assignment cases we investigated:

1) Identical unit capacities (no randomness; used as reference)

2) Integer randomly between 1 and 2

6) Integer randomly between 1 and 4

7) Integer randomly between 1 and 8

Cases 3 to 5 from before are replaced by Cases 6 and 7 here because we would like to study the effect of larger variances in the capacities.

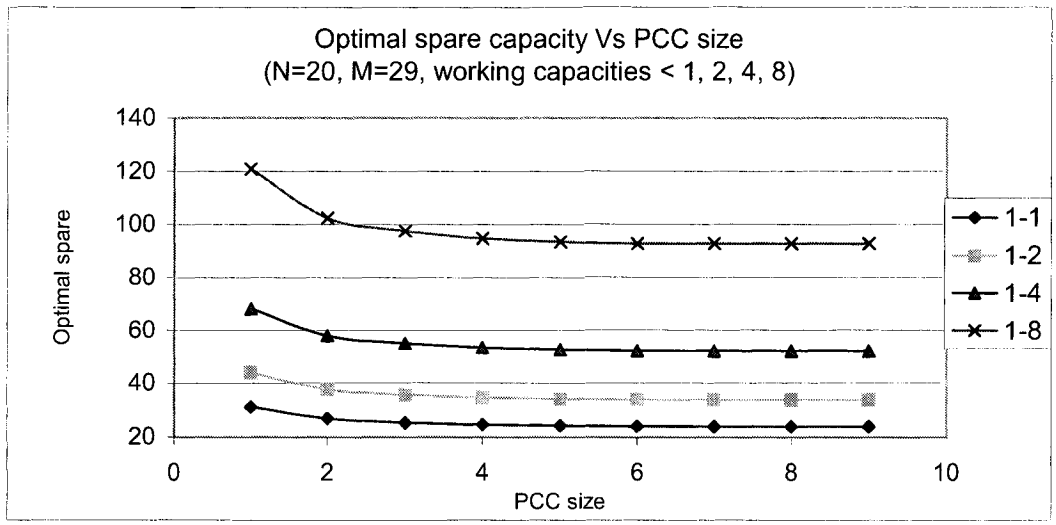


Figure 5.7: Optimal Spare Capacity with Respect to PCC Size

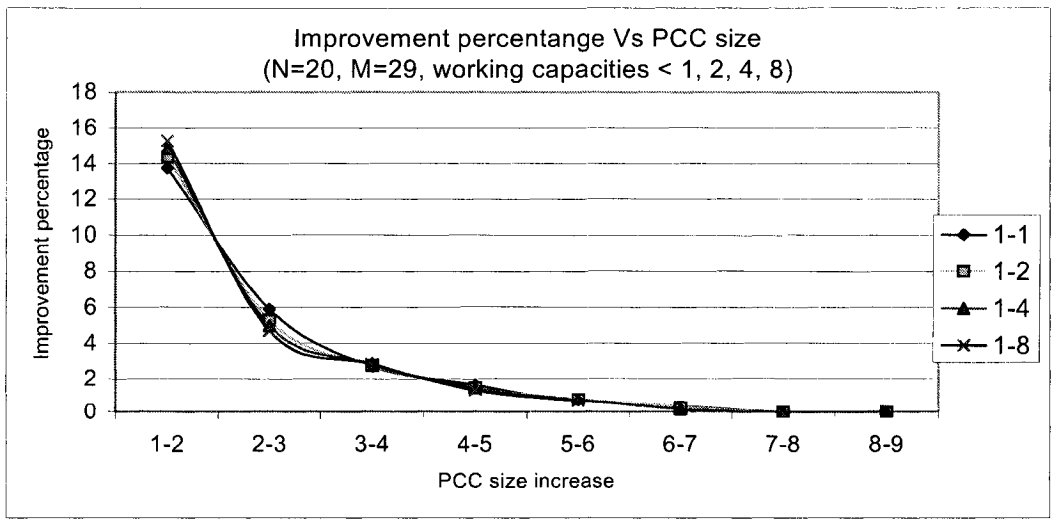


Figure 5.8: Improvement in Optimal Spare Capacity as a Percentage vs. the Increase in the Cycle Size in Terms of Faces

In Figure 5.7 we see that the four curves have similar behavior; a quick drop in the first few values then it shows kind of stability. The stability starts from around size 4 or 5. We mean by the stability point is the point after which the leveling of the curve is very small; i.e. does not increase or decrease as it was before that point. In the following figure (Figure 5.8), we compare the improvement percentage at each step. We can see that a similar behavior for the four curves; less that 4% improvement when increasing the face size more than four.

5.6 Searching Techniques

As seen from early discussion of BFS and DFS in Chapter 3 and their illustrations, the order of choosing the next neighbor nodes can result in different PCC set which is important in the evaluation of the resulted PCCs. Here, we shall investigate three different ordering techniques that can be combined with either the BFS or the DFS technique. These ordering can be used in increasing or decreasing order.

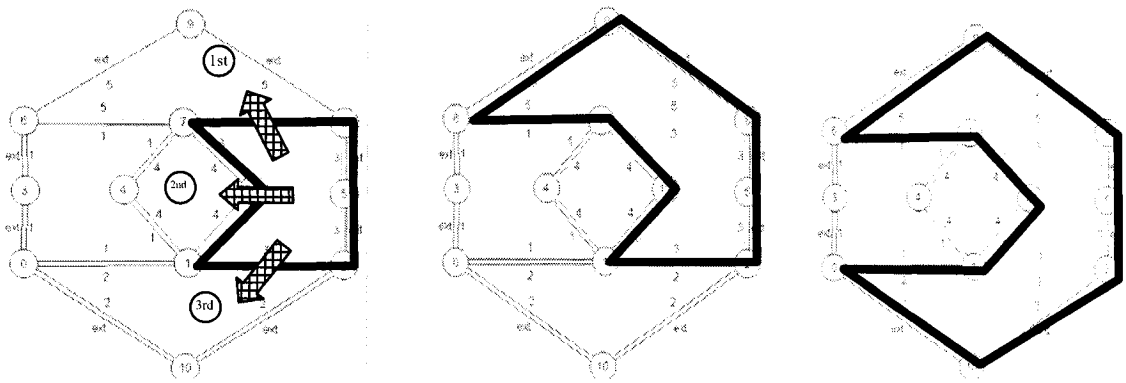
a) Label-based ordering: Since all faces are labeled, a simple way is to choose a face with the highest label (e.g. number) or the lowest label to start. This can be considered as a random way of choosing the next neighbor since the face labeling is assigned randomly.

b) Count-based ordering: This ordering is done by considering the number of unshared links between two faces. The more unshared links, the more on-cycle links there are in the resulting cycle. One can then choose a face with the most unshared links (choose one randomly if there is a tie). This is a good option for homogeneous networks where the links have identical capacities.

c) Cost-based ordering: This is done by considering the link installation costs on both faces and the shared links between them. A minimum is chosen from the minimum of three different parameters: the minimum link cost on both faces, and the minimum link cost on the shared edges between the two faces. The faces are then sorted (either high-to-low or low-to-high). This would be a good option in non-homogeneous networks where the links have different capacities on its edges.

Figure 5.9 is an example of Label-based ordering. Comparing to the example shown in Figure 3.15 notice how Face 5 (instead of 2) is chosen by this ordering in Step 1. in this particular example we have similar final result because of the few available choices as faces 4 and 1 were blocked to be inside the PCC, however the result is usually different as can be

easily expected.



Step-1 Step-2 Step-3

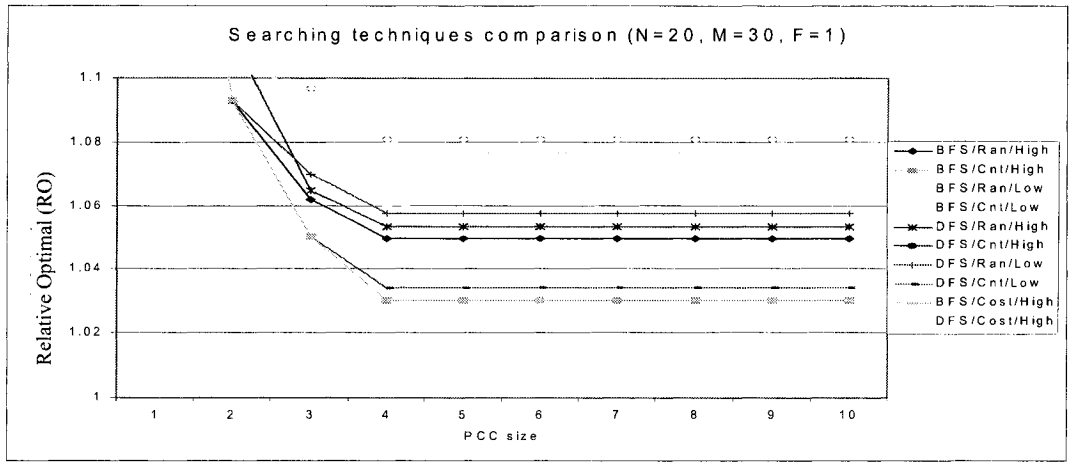
Figure 5.9: Running the Breadth First Search with Label High-To-Low Link Ordering Option

From various experiments, we have also determined that an increasing ordering or a decreasing ordering of the labels is not significant. However, this ordering is important in the second and the third option. More discussion on this effect will be shown on the Comparison subsection later on this chapter.

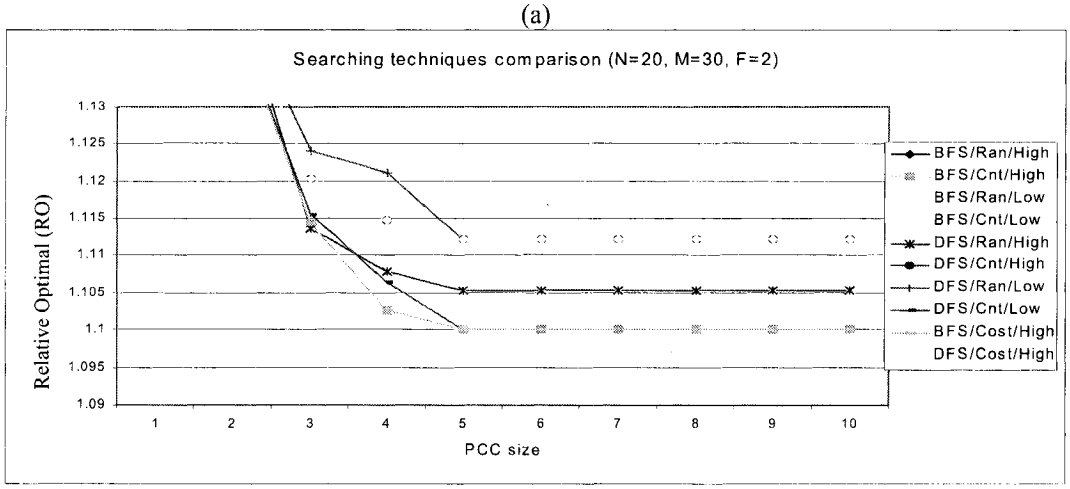
5.6.1 Comparing Different Searching Techniques

In order to compare the different ordering techniques discussed so far, we have conducted simulation on a large number of randomly-generated networks, each with different links capacities but with the same capacity distribution function for the working capacities. The simulations have also been repeated for different maximum to minimum ratio (denoted as F) in the link working capacities.

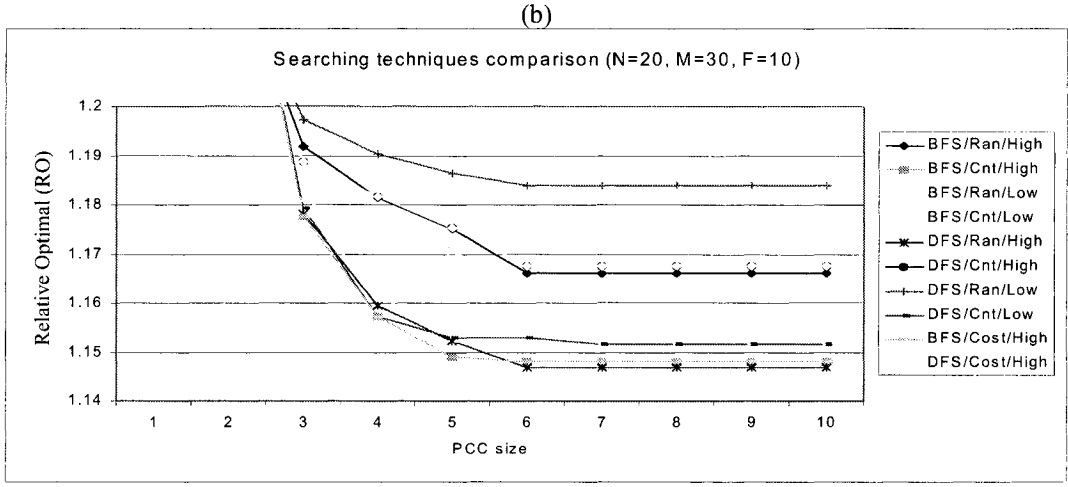
We would like to compare different (BFS/DFS, Label/Count/Cost, High/Low) combinations of search technique in terms of the relative optimal. We define the Relative Optimal (RO) as the ratio between the best solution (in spare capacities as in Section 5.5.2) that can be obtained by a given combination of search technique as given above and the best solution that can be obtained by the p-cycle method [ScSe05] when using all possible cycles within a given maximum protection path length. A low RO means the given method has a better performance since it is being compared to the best possible cycle-based method. In the following, we shall use High to indicate High-to-Low sorting while Low means Low-to-High



(a)



(b)



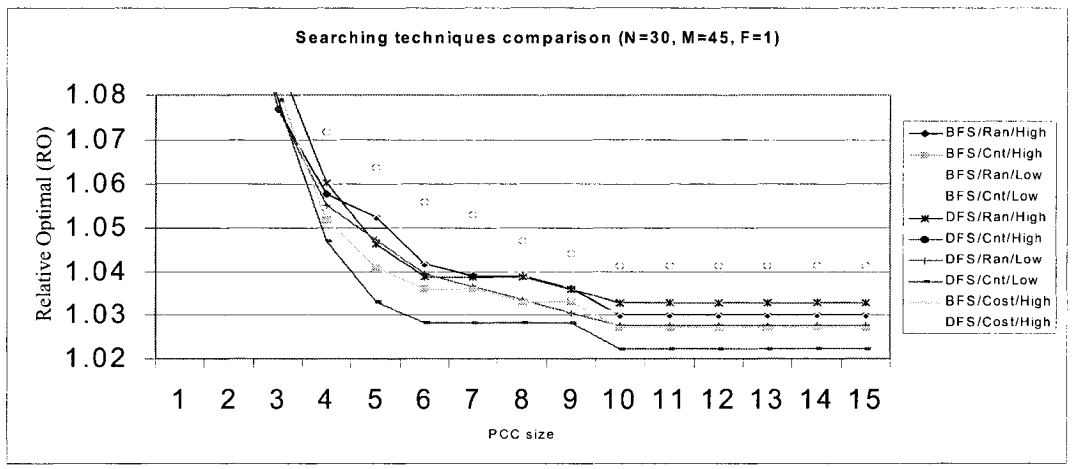
(c)

Figure 5.10: Comparing Different Searching Techniques (N=20, M=30) (a) F=1, (b) F=2 (c) F=10

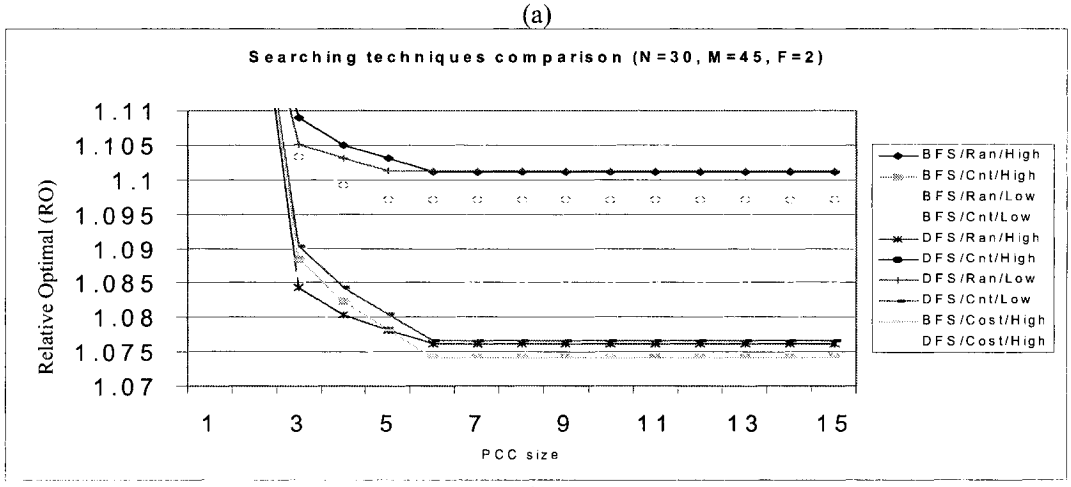
sorting. The denotation Ran/Cnt/Cost means the choice of label-based, count-based or cost-based respectively as detailed before.

Figure 5.10 shows the comparison for networks with N=20 nodes and M=30 edges but with a different F, each in a different plot. By examining Figure 5.10a (case of F=1), one can see that the RO performance curve of each technique is a decreasing function of the PCC size as before which levels off around PCC size of 4 faces.

By examining the other figures in Figure 5.10 we can see that (BFS, Cnt, High) technique appears to have the best (lowest) RO most of the time when F is low as the network is homogeneous in its capacities. On the other hand, (BFS, Cost, Low) appears to have the best performance in highly variant networks (high F value) since this ordering accounts for the cost.



(a)



(b)

Figure 5.11: Comparing Different Searching Techniques (N=30, M=45) (a) F=1, (b) F=2

In Figure 5.11 we show the comparison for larger network with 30 nodes and 45 edges. We can see similar general behavior by having (BFS, Cnt, High) as good searching technique

in such low capacity variant but large networks.

Form our various simulations, there does not appear to be a clear winning technique. One reason is that an optimization has been done on the different combinations to choose the best PCC set. However, we do observe that the (BFS, Cnt, High) technique appears to perform the best in homogeneous networks (networks with very little varying link capacity), while the (BFS, Cost, Low) technique can perform well in heterogeneous networks (networks with a high variation in link capacities).

Another important observation from studying the searching technique is to see that the PCC can get around 5%-15% less than the RO obtained from p-cycle using all possible cycles (all-cycles) which explained in 5.5.

This observation is important if computation time can be significant in obtain the optimal subset out of the candidate set. With a smaller candidate set and a polynomial complexity order (with respect to the network size), our algorithm has a big advantage when compared with the all-cycles method which has an exponential growth with respect to the network size.

5.7 Comparisons among Other Protection Methods

Before wrapping up our investigation of the protection application of the PCMCG algorithm, it would be of interest to see how it fares against the legacy approach of all-routs and the simple approach of ring protection. We do this by first comparing p-cycles against the all-rout and ring approaches using the RO measure defined in the last section. We then compare the number of PCC cycles generated by PCMCG algorithm with the total number p-cycles possible in the network under a given restriction on the protection path length. Obviously, the protection path length is the length of the path that is used to re-rout the traffic between two end nodes of a link that fails.

Appendix B gives an example to explain how the optimal solution is found for p-cycles, all-routes and ring works when a size restriction is imposed. In both p-cycle and rings a subset cycles with the given size restriction that have the minimal spare capacity is found to fully protect all the given working capacities. In all-routes, however, a subset of routes is found with the given length restriction that have the minimal spare capacity to protect all the working capacities.

5.7.1 Relative Optimum

We have conducted our experiments on 500 random graphs for different network sizes with a given max-to-min capacity F. The cost K units is used for F, i.e. $F=K=i$. The Relative Optimal (RO) is then measured in each case.

Figure 5.12 presents the average of all the measured RO as a function of the protection path length. We can see that solutions only exist for path length greater than or equal to 5. For example, the RO performance of rings at $F=1$ improves (reduces) as the protection path length increases a few more hops. It reaches its optimal at around 8 to 9 hops. P-cycles at $F=1$ start from a lower value but can only reach its best performance around 12 hops. Finally, the All-Routes scheme at $F=1$ starts from even a better value than the other two schemes and improves further as the path length increases only a few more hops. It stabilizes at around 10 hops length. The order of performance is All-routes (best), P-cycles and rings (worst). As F is increased from 1 to 4 and then to 10, the observations are similar.

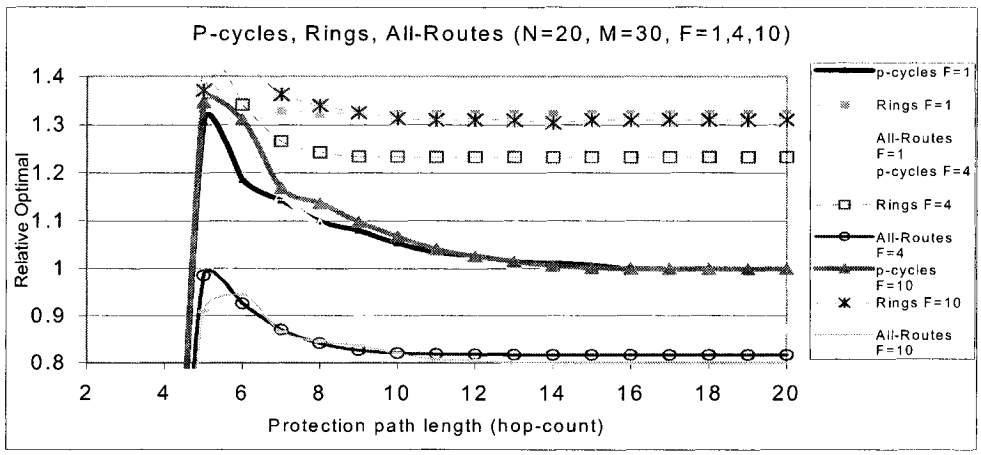


Figure 5.12: Comparison of Relative-Optimal Performance among Rings, P-Cycles and All-Routes (N=20, M=30, F=1,4,10)

Figure 5.13 shows the performance comparison for small networks of $N=8$ nodes and $M=12$ edges. Here similar observations are made except the stability points needs longer path to show up. Notice that the max possible path in such network is 8. Figure 5.14 shows the performance for a network with the same number of nodes and more edges (i.e. higher nodal degree) in which we see that the RO ratio increases which suggests that using short protection paths in highly variant working capacities network is not recommended since we will be much further from the global optimal.

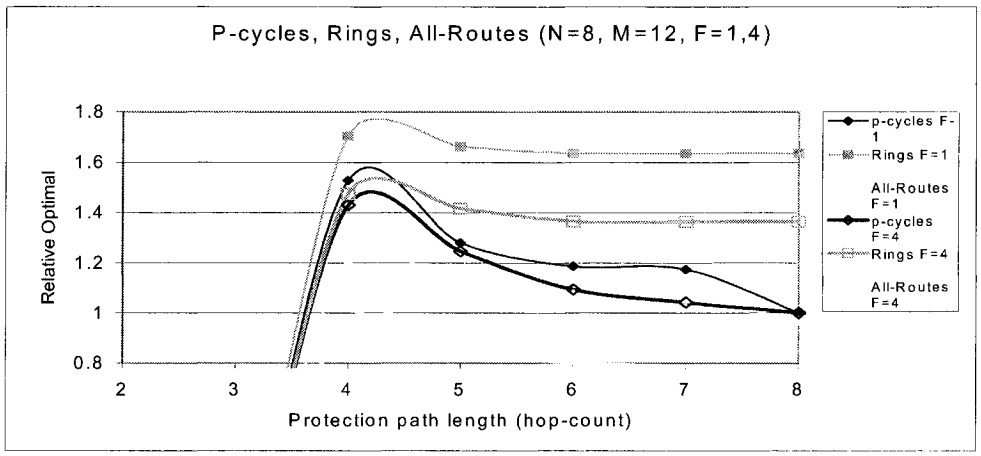


Figure 5.13: Comparison of Relative-Optimal Performance among Rings, P-Cycles and All-Routes (N=8, M=12, F=1,4)

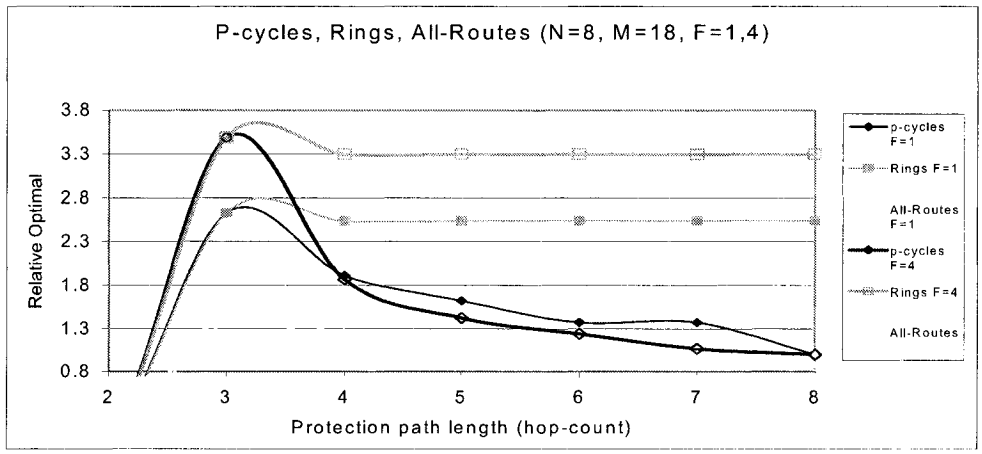


Figure 5.14: Comparison of Relative-Optimal Performance among Rings, P-Cycles and All-Routes (N=8, M=18, F=1,4)

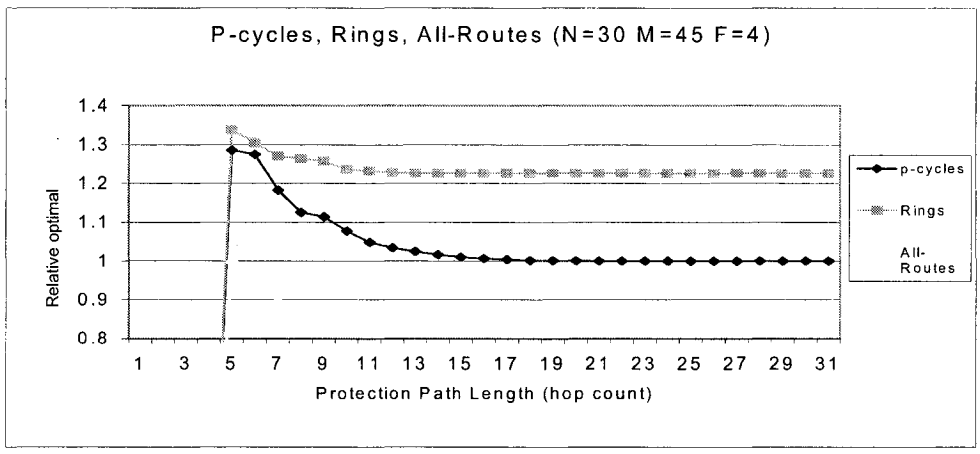
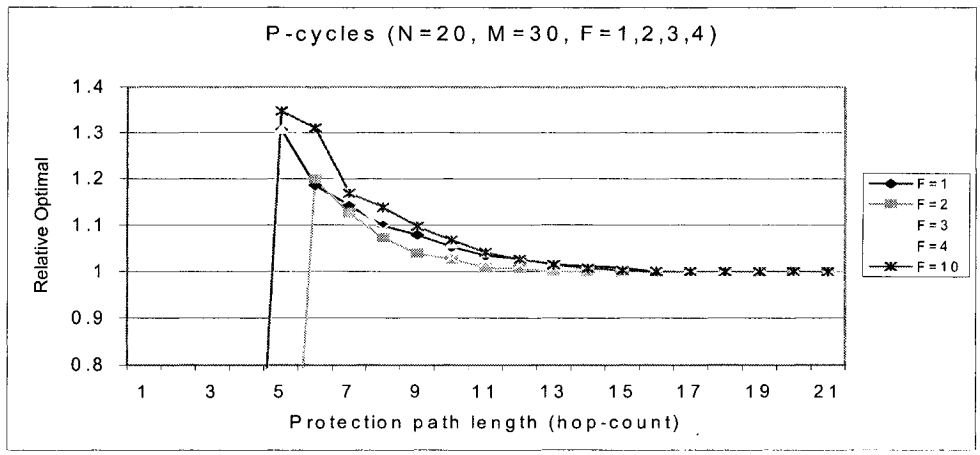
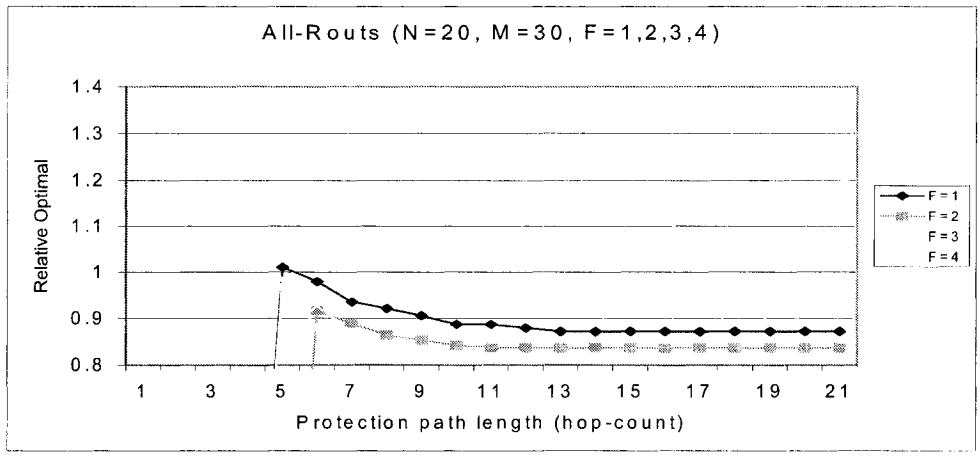


Figure 5.15: Comparison of Relative-Optimal Performance among Rings, P-Cycles and All-Routes (N=30, M=45, F=4)



(a)



(b)

Figure 5.16: Comparison of Relative-Optimal Performance among P-Cycles and All-Routes (N=20, M=30, F=1,2,3,4)

We can see that similar behavior stays the same for larger network with the same nodal degree except that the stability point noticed at shorter lengths and easier to decided. For example, in Figure 5.15 N= 30 nodes and M= 45 edges which have a nodal degree of 3 and thus it has the same nodal degree as Figure 5.13 but it is larger and the stability point show up at less path length compared to the maximum possible length for that graph. More examples are provided in Figure 5.16.

With the observations from other simulations, we can say in general that networks with a higher nodal degree can perform better (when restricting the size) under any protection scheme even with a small number of nodes (Ref: Figure 5.14).

If the protection capacity is relatively important, the choice will be between p-cycles and all-routes (i.e. ring is excluded). All-routes is better than p-cycle for around 30% when short

protection path lengths are used, but only about 10-20% for longer paths. This difference also depends on the F values (Ref: Figure 5.12 vs. Figure 5.13), and is only obvious when the networks are larger than 20 nodes and with an average nodal degree greater than 3. Under these scenarios, we need to only generate cycles or paths with length around half of the maximum size to reach a near optimal solution.

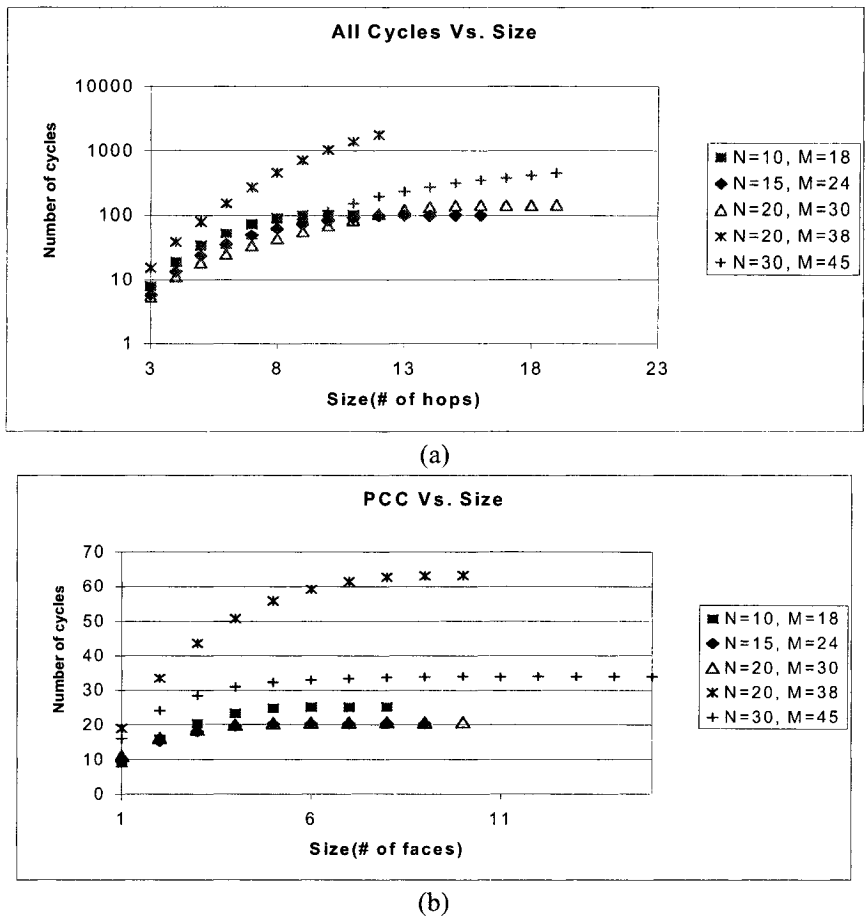


Figure 5.17: Comparison Between the Number of a) All Cycles and b) PCCs vs. Size for Different Networks

5.7.2 Number of Cycles and Cycle Size

As have been demonstrated before that a candidate set is used in an optimization model to find the best subset according to a given performance measure. The size this set and the complexity to find the optimal solution increases rapidly with the size of the candidate set and would require much longer computing time to solve. To investigate the effect/benefit of imposing a restriction on the protection path length and the size of the candidate set, we have used our PCMCG algorithm to find all the possible cycles when the number of internal faces

is restricted. Likewise, we do the same for the simple cycles (here called all-cycles) which is used in p-cycle. However, in this case the size is the number of edges.

A simple comparison can be observed by noticing that the cycle size in term of faces has the range $[1,F]$ whereas the range of the cycle size in term of edges is $[3,N]$. Let $Mf= (N-3)/(F-1)$ which represents the ratio of the cycle (in term of edges) to the cycle size (in term of faces). Since $F=M-N+1$, one obtains $Mf=(N-3)/(M-N)$. Then if $S1$ is the size of a cycle in term of faces and $S2$ is the size of the cycle in term of edges, we have on the average $S2=S1*Mf+2$. For example for a network with N=10, M=18, Mf=9/8=1.125, then for cycles of size 2 faces, then, in average, it consists of 4 edges. Consider an example network with N=10, M=18, Mf=9/8=1.125. The cycles would have 4 edges on the average if each cycle has 2 faces on the average.

Figure 5.17a depicts the number of cycles generated as a function of size limit (in terms of the number of hops of a cycle) using the all-cycles method, while Figure 5.17b gives the number of PCCs generated with respect to the size limit measured in the number of faces. Both figures have performances for various networks sizes (N,M). Note that Figure 5.17a is plotted on a log scale. Also the size limit can only increase to as many as N.

For a network with N=10 nodes and M=18 edges in Figure 5.17a, the number of cycles generated is an increasing function of size limit but levels off at around 100. However this number increases rapidly as the network size increases, e.g. to around 500 for a network with N=30 nodes and M=45 edges .

On the other hand, we can see that in Figure 5.17b, the number of cycles generated is an increasing function of size limit but levels off at around 25. This number increases as the network size increases in a reasonable ratio, e.g. to around 35 for a network with N=30 nodes and M=45 edges .

One can easily observe that the number of PCCs for N=10 and M=18 increases from around 8 (when the cycle size is 1) and levels off to around 25 (when the cycle size is 6), whereas the number of all-cycles starts increasing exponentially from 8 at size 3 up to 100 at size 10. For larger network sizes, N=20 nodes and M=38 edges, where we have almost doubled the number of nodes and edges, the number of PCC is has increases by a factor of 2 to 3 while the number of all-cycles have increased by a factor of more than 15.

From our experiments, we can say that the "stability" point (defined in Section 5.5) of PCCs is at about half of the max possible size. Whereas for all-cycles is at about 70-80% of the max possible size.

It is interesting to note that Figure 5.17 also provides the information on the effect of nodal degree. For example, a network of N=20 nodes and M=30 links would give a nodal degree of 3. From the figures, we can see that when the nodal degree of the node is increased from 3 to 3.8 (i.e. networks with N=20 nodes and M=38 links) the number of all cycles increased from by a factor of 10 or more, while the number of PCCs increases within a factor of 3. Other network examples have been provided for completeness.

5.8 Concluding Remarks

We have presented in this chapter the approaches of using PCC size limit, PCC coverage and the different choices of picking neighboring nodes (and therefore modifications of our original MCG algorithms) to find PCCs efficiently for network protection. Through the performance evaluations of these techniques (using both practical and random networks) as well as comparing with other protection techniques, we have demonstrated the advantages of PCCs in providing cycle coverage and achieving a good level of optimality.

Since we show that our PCC is better than p-cycle in term of the candidate set and no much difference in the optimal solution achieved, this demonstrates the efficiency of PCC.

Chapter Six
PCC-based Protection Analysis

In this chapter we will provide an analysis for the network restorability and some other performance measures for cycle-based protection schemes in mesh optical networks. Specifically, we want to study the use of Hamiltonian cycles for protection purpose, not only because Hamiltonian cycle is a special case of many cycle-based algorithms such as p-cycle and our PCC, but because they can be generated by our MCG algorithm and have been partially analyzed in Chapter Five. Here, we shall first provide the theoretical analysis of the single cycle case which has attracted much research as discussed in the Literature Review in Chapter One which will serve as a bench mark. We then endeavor to tackle the difficult task of analyzing the Hamiltonian cycles in two domains that would provide insight into multiple domain analysis. To represent more possible scenarios of future network and verify the proposed analysis, we have used random graphs that are generated with conditions that approximate realistic networks. Simulation study is also presented for comparison with the theoretical study. We hope such exercise can help us to forecast/plan for the network performance when using cycle-based network protection techniques, and to shed lights into finding more efficient techniques or possible improvements for link protection mechanisms in mesh networks.

6.1 Domain Model and Definitions

Recall from Chapter Two that a Hamiltonian cycle is a simple cycle that traverses all the network nodes. Unfortunately, not every graph is a Hamiltonian cycle, and our protection operation in Section 5.1 shows that more than one cycle (Hamiltonian and non-Hamiltonian) are required to cover all nodes in a general planar network. In order to develop an analytical model and to perform optimization later for various performance measure (such as restorability), we shall define D-domain coverage as the process of using D Hamiltonian cycles to cover all nodes. This is done in such a way that each domain has a clear cut boundary with its neighboring domain and that each domain is protected by only one Hamiltonian cycle. For example, Section 5.4.3 has discussed how a number of PCCs were

generated to cover the network. Each PCC in that section could be thought of as one domain. We shall provide the restorability and other performance measures such as redundancy and protection path length analysis of this type of coverage in the sections coming up.

Restorability is an important performance parameter to measure how efficient a protection scheme uses network resources. Let us first define the *total network restorability* as the percentage of the total working capacity that could be restored in case of any single link failure, for a given amount of protection capacity in the network. For a pre-designed fixed capacity network with a known amount of working capacity on each link, one objective is to determine how much of the network working capacity (from all the links) can be protected by the remaining capacity on each link (i.e. the available protection capacity) in case of any single link failure in the network. Another important performance measure that will be used and analyzed later on is the *network redundancy* (R_d) which is defined as the ratio of the protection capacity to the working capacity of the network in order to achieve maximum restorability (100%) for a given network.

We define the *protected working capacity* c_{wpi} on a link as the amount of link working capacity that can be protected by the protection cycle (Hamiltonian in the analysis in this Chapter), and the *unprotected working capacity* c_{wui} on a link as the part that has no protection because there is not enough protection capacity contributed by the cycle protection path. We can likewise define the following parameters: C_{pavg} and C_{uavg} as the average of both protected and unprotected capacities, C_{wavg} as the average working capacities on the links. All of the parameters above are average among all links of the same network. The total network restorability (will simply be referred to as restorability if there is no ambiguity in meaning) can thus be calculated as following:

$$R = \frac{\sum_{i=1}^{m} C_{wpi}}{\sum_{i=1}^{m} C_{wi}} = \frac{M \times C_{pavg}}{M \times C_{wavg}} = \frac{C_{wavg} - C_{uavg}}{C_{wavg}} = 1 - \frac{C_{uavg}}{C_{wavg}} \tag{6.1}$$

Every link on the cycle should be able to protect either a failed on-cycle link or a straddling link of the cycle. Consequently, there are couples of considerations/observations. 1) Since the amount of protection capacity on a cycle is determined by the weakest (lowest

capacity) link, the restorability of a protection cycle in a fixed capacity network is constrained by that minimum value when that link is an on-cycle link. 2) A failed straddling link can have the following three operation models according whether it will/can use the information from one or two paths and the consideration would affect the restorability analysis.

The RBP (Random Backup Path) Model: When a straddling link fails, it will re-route its traffic along one of the two backup paths it selects randomly.

The SBP (Shorter Backup Path) Model: When a straddling link fails, it will re-route its traffic along the shorter of the two backup paths. Since there are only two paths, the shorter path is also the shortest path.

The TBP (Two Backup Path) Model: When a straddling link fails, it will re-route its traffic along both backup paths.

Unlike the TBP, both RBP and SBP use only one backup path.

The following assumptions are used in our analysis that follows:

1. All links are bidirectional.
2. Each node is connected to at least two other nodes (i.e., a node degree of more than or equal to two).
3. The network can be decomposed into D Hamiltonian subgraphs.
4. Each subgraph is protected by a Hamiltonian cycle.
5. Each cycle link is protected by a backup path that traverses the remaining links of the cycle between the two end-nodes.
6. The working capacity on each link is randomly distributed according to a uniformly distributed function f_{cw} bounded by c_{wmin} and c_{wmax}.
7. The protection capacity on each link is randomly distributed according to a uniformly distributed function f_{cp} bounded by c_{pmin} and c_{pmax}.

The last two assumptions on uniform distribution with fixed bounds make our analysis more realistic than a pure random analysis. Moreover, we have observed the existence of such bounds from our simulations; the uniform distribution is relaxed for simplicity.

6.2 Single Domain Restorability Analysis

For single-domain (D=1) coverage, we assume that the network has already been covered by

a single Hamiltonian protection cycle. We first provide the common analysis, and then the restorability is analyzed using all three operation models for a failed chordal link.

Let X be a random variable for the minimum available protection capacity on a protection path. For a given value of $X=x$, the amount of unprotected working capacity on an arbitrary link e_i can be computed as follows:

$$c_{wui} = \text{Max}(0, c_{wi} - x) = u_i \times (c_{wi} - x),$$
$$\text{where } u_i = \begin{cases} 0 & c_{wi} \le x \\ 1 & c_{wi} > x \end{cases} \tag{6.2}$$

Let $C_{uavg}(x)$ be the average unprotected working capacity conditioned on a minimum protection capacity x of the protection path, and f_{cw} be the probability density function for an amount of working capacity x on a link. We then have

$$C_{uavg}(x) = \int_{c_{w\min}}^{c_{w\max}} u \times (y - x) f_{cw}(y) dy, \qquad u = \begin{cases} 0 & y \le x \\ 1 & y > x \end{cases}$$
$$C_{uavg}(x) = \int_{x}^{c_{w\max}} (y - x) f_{cw}(y) dy \tag{6.3}$$

The value of x depends also on the length of the protection path. As explained before, we need to consider two link cases here: the cycle links and the straddling links. Let $f_{cp}(x)$) be the density function of an amount x of link protection capacity, and $F_{cp}(x)$ the cumulative distribution function of f_{cp}. Let $f_X(x \mid k)$ be the conditional density function of x on a given protection path of k hops. Then we have

$$f_X(x \mid \text{PathLength}= k) = \sum_{1}^{k} f_{cp}(x) \times Pr(\text{(k-1) links have protection capacities larger than } x)$$

$$= \sum_{1}^{k} f_{cp}(x) \times (1 - F_{cp}(x))^{k-1} = k f_{cp}(x) \times (1 - F_{cp}(x))^{k-1} \tag{6.4}$$

Using the same upper and lower bounds for both working and protection capacity, we have: $c_{wmax} = c_{pmax} = c_{max}$ and $c_{wmin} = c_{pmin} = c_{min}$, Then we have on average $C_{wavg} = (c_{max} + c_{max})/2$. The probability density $f_{cp}(x)$ and the cumulative function $F_{cp}(x)$ of the protection capacity are

$$f_{cp}(x) = f_{cw}(x) = \frac{1}{c_{\max} - c_{\min}}$$

$$F_{cp}(x) = F_{cw}(x) = \frac{x - c_{\min}}{c_{\max} - c_{\min}} \tag{6.5}$$

Substituting (6.5) into (6.4), we have

$$f_X(x \mid k) = k \times \frac{1}{c_{\max} - c_{\min}} \times \left(\frac{c_{\max} - x}{c_{\max} - c_{\min}} \right)^{k-1} = k \frac{(c_{\max} - x)^{k-1}}{(c_{\max} - c_{\min})^k} \tag{6.6}$$

We can also substitute (6.5) into (6.3) to calculate the conditional average unprotected working capacity as following:

$$C_{uavg}(x) = \int_x^{c_{\max}} (y - x) \frac{1}{c_{\max} - c_{\min}} dy = \frac{(c_{\max} - x)^2}{2(c_{\max} - c_{\min})} \tag{6.7}$$

Since there is only one Hamiltonian cycle covering the network of N nodes and M links, each network link could be either a cycle link with a probability of $\frac{N}{M}$, or a straddling link with a probability of $\frac{M - N}{M}$. Using the un-conditioning techniques, we obtain

$$\begin{aligned} f_X(x) = {} & f_X(x \mid \text{cycle}) \times \Pr\{\text{cycle}\} + \\ & f_X(x \mid \text{Straddling}) \times \Pr\{\text{Straddling}\} \end{aligned} \tag{6.8}$$

We are now ready to complete the restorability analysis of the Random Backup Path Model and the Shortest Backup Path Model in the following.

6.2.1 The Random Backup Path Model

Under this model, whenever a straddling link fails we randomly choose one of the two paths provided by the Hamiltonian cycle to re-route the traffic.

For every cycle link, the protection path length is *N-1*. As for a straddling link, the minimum path between two nodes (other than the directly connected straddling link) obviously has to be at least two links long. Therefore, two is the minimum value for the protection path length for a failed straddling length regardless of which one of the two paths

is chosen. Since the minimum for the first protection path is 2, the maximum of the second (alternate) protection path (on the opposite direction) will be *N-2* hops long (so that the total length of a Hamiltonian cycle is always *N*). Since any one of the two protection paths can be chosen randomly, the path length k has a range of $2 \le k \le N-2$. For a uniform length distribution, we have

$$\Pr(\text{path length} = \text{k}) = \frac{1}{((N-2)-2)+1} = \frac{1}{N-3} \tag{6.9}$$

Note that the expression is independent of k, because there is an equal probability of picking a backup path of any length within the range. Continuing on from (6.8), we have

$$f_X(x) = \sum_k f_x(x|k) \times \Pr(\text{Path Length} = \text{k})$$
$$= \frac{N}{M}\left[(N-1) \times f_{cp}(x)\left(1-F_{cp}(x)\right)^{N-2}\right] + \frac{M-N}{N}\left[\sum_{k=2}^{N-2} \frac{1}{N-3} k f_{cp}(x)\left(1-F_{cp}(x)\right)^{k}\right]$$

By using (6.6) and the fact that the number of the straddling and the on-cycle links are (*M-N*) and *N*, respectively. We have

$$f_X(x) = \frac{N(N-1)}{M} \times \frac{(c_{\max}-x)^{N-2}}{(c_{\max}-c_{\min})^{N-1}} + \frac{M-N}{M(N-3)} \sum_2^{N-2} k \frac{(c_{\max}-x)^{k-1}}{(c_{\max}-c_{\min})^{k}} \tag{6.10}$$

Use (6.10) to remove the condition in (6.7), and with a change of variable $v = \frac{c_{\max}-x}{c_{\max}-c_{\min}}$, the average unprotected working capacity can now be computed as following for the random choice scenario:

$$C_{uavg} = \int_{c_{p\min}}^{c_{p\max}} \left(\int_{c_{w\min}}^{c_{w\max}} c_{wui} f_{cw}(y) dy \right) f_X(x) dx = \int_{c_{p\min}}^{c_{p\max}} \left(\frac{1}{c_{\max}-c_{\min}} \int_{c_{w\min}}^{c_{w\max}} u_i \times (c_{wi}-x) dy \right) f_X(x) dx$$

Then

$$C_{uavg} = \int_0^1 \frac{v^2}{2} \times \left(\frac{N(N-1)}{M} v^{N-2} + \frac{M-N}{M(N-3)} \sum_2^{N-2} k v^{k-1} \right) \times (c_{\max} - c_{\min}) dv$$

$$= \frac{c_{\max} - c_{\min}}{2} \left(\int_0^1 \frac{N(N-1)}{M} v^N dv + \frac{M-N}{M(N-3)} \sum_2^{N-2} k \int_0^1 v^{k+1} dv \right) \tag{6.11}$$

$$= \frac{c_{\max} - c_{\min}}{2} \left(\frac{N(N-1)}{M(N+1)} + \frac{M-N}{M(N-3)} \sum_2^{N-2} \frac{k}{k+2} \right)$$

Substituting (6.11) in (6.1), we can compute the restorability R for the random choice case as following:

$$R(\text{Random Choice}) = 1 - \frac{F-1}{F+1} \left(\frac{N(N-1)}{M(N+1)} + \frac{M-N}{M(N-3)} \sum_2^{N-2} \frac{k}{k+2} \right) \tag{6.12}$$

where $F = \frac{c_{\max}}{c_{\min}}$ is the ratio of maximum to minimum link working capacity in the network.

One can see that F plays the most important role in the restorability formula, and clearly shows how the restorability of cycle schemes decrease in networks with unequal link capacities. This is also the case in the other models.

6.2.2 The Shortest Backup Path Model

To recap, by the shortest backup path, we mean that whenever a straddling link fails we choose the shorter of the two paths provided by the Hamiltonian cycle to protect this link.

Using this model, we have $k_{min}=2$ *and* $k_{max}= N/2$, i.e. $2 \le k \le \frac{N}{2}$, Again, based on uniform distribution $\Pr(\text{path length} = k) = \left((N/2) - 2 + 1\right)^{-1} = \frac{2}{N-2}$. Starting with (6.8) and using a similar approach as in the previous section, we have

$$R(\text{Shorter Cycle Path}) = 1 - \frac{F-1}{F+1} \left(\frac{N(N-1)}{M(N+1)} + \frac{2(M-N)}{M(N-2)} \sum_2^{\frac{N}{2}} \frac{k}{k+2} \right) \tag{6.13}$$

Again the factor F stands out. As a simple verification note that for $F=1$ (a flat network, where link capacities are equal everywhere) the maximum achievable restorability is one. As pointed out before, the existence of a Hamiltonian cycle in this case is enough to guarantee

full restorability in a flat network, because every protection path is capable of handling the working traffic of any other links in the network. When F becomes big as in a hierarchical or an arbitrary network, the restorability R of the Hamiltonian cycle becomes very low.

6.2.3 Approximations

The formulas presented in the previous two subsections could be approximated (and further simplified) if we assume the straddling protection path length is similar to the on-cycle protection path length. In such case, the length of every backup path (cycle or straddling) is *N-1* hops long, and we can write

$$\begin{aligned} f_X(x) &\approx (N-1)\times f_{cp}(x)\left[1-F_{cp}(x)\right]^{N-2} \\ &= (N-1)\times\frac{1}{c_{\max}-c_{\min}}\times\left(\frac{c_{\max}-x}{c_{\max}-c_{\min}}\right)^{N-2} \\ &= \frac{(N-1)(c_{\max}-x)^{N-2}}{(c_{\max}-c_{\min})^{N-1}} \end{aligned}$$

$$C_{wu\_avg} \approx \int_{c_{\min}}^{c_{\max}} \frac{(c_{\max}-x)^2}{2(c_{\max}-c_{\min})}\times\frac{(N-1)(c_{\max}-x)^{N-2}}{(c_{\max}-c_{\min})^{N-1}}dx = \frac{1}{2}\int_{c_{\min}}^{c_{\max}}\frac{(N-1)(c_{\max}-x)^N}{(c_{\max}-c_{\min})^N}a$$

$$\Rightarrow C_{wu\_avg} \approx \frac{1}{2}(c_{\max}-c_{\min})\frac{N-1}{N+1} \tag{6.14}$$

Therefore,

$$R \approx 1-\frac{F-1}{F+1}\times\frac{N-1}{N+1}, \quad F=\frac{c_{\max}}{c_{\min}} \tag{6.15}$$

This very interesting result indicates that in a mesh network with uniformly distributed pre-determined link capacities, the total network restorability using a Hamiltonian protection cycle is a decreasing function of both total number of nodes N, and the ratio of maximum to minimum link capacity, F. As expected for a flat network (F=1) where link capacities are identical, the restorability is equal to one. It also sets the lower bound for restorability to $\frac{2}{N+1}$, for the case where the ratio of maximum to minimum link capacity is large.

The above approximation hides the fact that the restorability in a network does indeed change slightly with the average nodal degree of the network as revealed by Sections 6.2.1

and 6.2.2, which is the subject of more discussion in the next section. However, it provides a quick, rough, and lower bound estimation.

6.2.4 Both Backup Paths Model

When both backup paths are used, traffic on a failed straddling link can be re-routed along both paths simultaneously. Therefore, the required protected working capacity on a straddling link can be shared by the protection capacity for the two paths in the Hamiltonian cycle. Let us denote the minimum available protection capacity on a protection path provided by a protection cycle by a continuous random variable X. For a given value of $X=x$, the amount of unprotected working capacity on an arbitrary link e_i can be calculated as follows:

$$c_{wui} = \text{Max}(0, c_{wi} - x) = u_i \times (c_{wi} - x), \quad i \quad on-cycle \quad u_i = \begin{cases} 0 & c_{wi} \le x \\ 1 & c_{wi} > x \end{cases}$$

$$c_{wui} = \text{Max}(0, c_{wi} - x - y) = u_i \times (c_{wi} - x - y), \quad i \quad straddling, \quad u_i = \begin{cases} 0 & c_{wi} \le x + y \\ 1 & c_{wi} > x + y \end{cases}$$

where y is the minimum protection capacity from the alternate path. Without loss of generality assume $x \le y$. Then in the worst case (least restorability of the straddling link) y will be equal to x. Then for any straddling link i, the total average not protected working capacity is the maximum of 0 or $c_{wi} - 2x$. It follows that for any link i

$$c_{wui} = \begin{cases} \text{Max}(0, c_{wi} - x) = u_{1i} \times (c_{wi} - x), & i \text{ on-cycle} \\ \text{Max}(0, c_{wi} - 2x) = u_{2i} \times (c_{wi} - 2x), & i \text{ straddling} \end{cases}$$

where $u_{1i} = \begin{cases} 0 & c_{wi} \le x \\ 1 & c_{wi} > x \end{cases}$ and $u_{2i} = \begin{cases} 0 & c_{wi} \le 2x \\ 1 & c_{wi} > 2x \end{cases}$

As discussed in the previous sections, for a network with N nodes and M links protected by one Hamiltonian cycle, the cycle length is N hops long, and each network link is either a cycle link with a probability of $\frac{N}{M}$, or a straddling link with a probability of $\frac{M-N}{M}$. Thus

$$C_{uavg}(x) = \frac{N}{M}\int_{x}^{c_{w\max}} (z-x) f_{cw}(z)dz + \frac{M-N}{M}\int_{2x}^{c_{w\max}} (z-2x) f_{cw}(z)dz \tag{6.16}$$

where $C_{uavg}(x)$ represents the value of average unprotected working capacity conditioned on the minimum protection capacity on the protection path, and f_{cw} is the probability density function for the working capacity on a link. The value of x, which is the minimum protection capacity on a protection path, depends also on the length of the protection path. Let $f_{cp}(x)$ be the density function for an amount of x link protection capacity, and $F_{cp}(x)$ the cumulative distribution function. Then $f_X(x \mid k)$, the conditional density function of x conditioned on a given protection path with a length of k hops can be obtained as follows.

$$f_X(x \mid \text{Path Length}=k) = \sum_{1}^{k} f_{cp}(x) \times \textit{Pr((k-1) links have protection capacities larger than x)}$$

$$= \sum_{1}^{k} f_{cp}(x) \times (1-F_{cp}(x))^{k-1} = k\, f_{cp}(x) \times (1-F_{cp}(x))^{k-1} \tag{6.17}$$

Again, if we consider that both the working and the protection capacities of the network links are uniformly distributed random variables, and have identical probability density functions, we have

$c_{wmax} = c_{pmax} = c_{max}$ and $c_{wmin} = c_{pmin} = c_{min}$ and $c_{wavg} = \frac{c_{\max} + c_{\min}}{2}$

$$f_{cp}(x) = f_{cw}(x) = \frac{1}{c_{\max} - c_{\min}} \quad \text{and} \quad F_{cp}(x) = F_{cw}(x) = \frac{x - c_{\min}}{c_{\max} - c_{\min}}$$

$$f_{cp}(x) = f_{cw}(x) = \frac{1}{c_{\max} - c_{\min}} \quad \text{and} \quad F_{cp}(x) = F_{cw}(x) = \frac{x - c_{\min}}{c_{\max} - c_{\min}} \tag{6.18}$$

Substituting from (6.18) into (6.17), we will have:

$$f_X(x \mid k) = k \times \frac{1}{c_{\max} - c_{\min}} \times \left[\frac{x - c_{\min}}{c_{\max} - c_{\min}}\right]^{k-1} = k \frac{(c_{\max} - x)^{k-1}}{(c_{\max} - c_{\min})^{k}} \tag{6.19}$$

By substituting (6.18)) into (6.16), we obtain the conditional average unprotected working capacity as following:

$$C_{uavg}(x) = \frac{N}{M}\int_{x}^{c_{w\max}} (z-x)\frac{1}{c_{\max}-c_{\min}}dz + \frac{M-N}{M}\int_{2x}^{c_{w\max}} (z-2x)\frac{1}{c_{\max}-c_{\min}}dz$$

$$= \frac{N}{M}\cdot\frac{(c_{\max}-x)^2}{2(c_{\max}-c_{\min})} + \frac{M-N}{M}\cdot\frac{(c_{\max}-2x)^2}{2(c_{\max}-c_{\min})}$$

$$= \frac{1}{2M}\left(N\frac{(c_{\max}-x)^2}{(c_{\max}-c_{\min})} + (M-N)\frac{(c_{\max}-2x)^2}{(c_{\max}-c_{\min})}\right)$$

Then

$$C_{uavg}(x) = \frac{l}{2M}\left(Nv^2 + (M-N)\left(4v^2 - \frac{4vc}{l} + \frac{c^2}{l^2}\right)\right) \tag{6.20}$$

where $v = \frac{c_{\max}-x}{c_{\max}-c_{\min}}, l = c_{\max}-c_{\min}$.

It has been shown in Section 6.2.1 that the length k of the backup paths of a straddling link has a range of $2 \le k \le N\text{-}2$. Using the un-conditioning technique and totality of probability, we can remove the condition on k to obtain the following:

$$f_X(x \mid \text{two-paths-used}) = f_X(x \mid \text{cycle}) \times \Pr\{\text{cycle}\} + f_X(x \mid \text{Straddling}) \times \Pr\{\text{Straddling}\}$$

$$= \frac{N}{M} f_X(x \mid k = N-1) + \frac{M-N}{M} f_X(x \mid k = N)$$

$$= \frac{N(N-1)}{M} \times \frac{(c_{\max}-x)^{N-2}}{(c_{\max}-c_{\min})^{N-1}} + \frac{(M-N)N}{M}\frac{(c_{\max}-x)^{N-1}}{(c_{\max}-c_{\min})^{N}}$$

$$= \frac{N}{lM}\left[(N-1)\times v^{N-2} + (M-N)v^{N-1}\right] \tag{6.21}$$

Using (6.21) to remove the condition in (6.20), we obtain the average unprotected working capacity can now be computed as follows.

$$C_{uavg} = \frac{l}{2M}\frac{N}{lM} l \int_0^1 \left(\left(Nv^2 + (M-N)\left(4v^2 - \frac{4vc}{l} + \frac{c^2}{l^2}\right)\right)\left[(N-1)\times v^{N-2} + (M-N)v^{N-1}\right]\right) dv$$

$$C_{uavg} = \frac{Nl}{2M^2}\int_0^1 \Big(\Big[(N-1)\times v^{N-2} + (M-N)v^{N-1} \Big] Nv^2 + \Big[(N-1)\times v^{N-2} + (M-N)v^{N-1} \Big](M-N)\left(4v^2 - \frac{4vc}{l} + \frac{c^2}{l^2} \right) \Big) dv$$

$$Cavg = \frac{Nl}{2M^2}\int_0^1 \Big(N(N-1)\times v^N + N(M-N)v^{N+1} + (M-N)\Big(4v^2\Big[(N-1)\times v^{N-2} + (M-N)v^{N-1}\Big] - \frac{4vc}{l}\Big[(N-1)\times v^{N-2} + (M-N)v^{N-1}\Big] + \frac{c^2}{l^2}\Big[(N-1)\times v^{N-2} + (M-N)v^{N-1}\Big]\Big)\Big) dv$$

$$Cavg = \frac{Nl}{2M^2}\int_0^1 \Big(N(N-1)\times v^N + N(M-N)v^{N+1} + (M-N)\Big[4\Big[(N-1)\times v^N + (M-N)v^{N+1}\Big] - \frac{4c}{l}\Big[(N-1)\times v^{N-1} + (M-N)v^N\Big] + \frac{c^2}{l^2}\Big[(N-1)\times v^{N-2} + (M-N)v^{N-1}\Big]\Big]\Big) dv$$

$$Cavg = \frac{Nl}{2M^2}\Big(N\left(\frac{(N-1)}{N+1} + \frac{(M-N)}{N+2}\right) + (M-N)\Big[4\left[\frac{(N-1)}{N+1} + \frac{(M-N)}{N+2}\right] - \frac{4c}{l}\left[\frac{(N-1)}{N} + \frac{(M-N)}{N+1}\right] + \frac{c^2}{l^2}\left[1 + \frac{(M-N)}{N}\right]\Big]\Big)$$

Finally,

$$R = 1 - \frac{Nl}{LPM^2}\Big(\left[N\left(\frac{(N-1)}{N+1} + \frac{(M-N)}{N+2}\right)\right] + (M-N)\Big[4\left[\frac{(N-1)}{N+1} + \frac{(M-N)}{N+2}\right] - \frac{4c}{l}\left[\frac{(N-1)}{N} + \frac{(M-N)}{N+1}\right] + \frac{c^2}{l^2}\left[1 + \frac{(M-N)}{N}\right]\Big]\Big) \tag{6.22}$$

6.2.5 Simulations and Verifications

We first present a network example adapted from a real network to demonstrate the applicability of our analysis and to verify the accuracy of our analytical formulas.

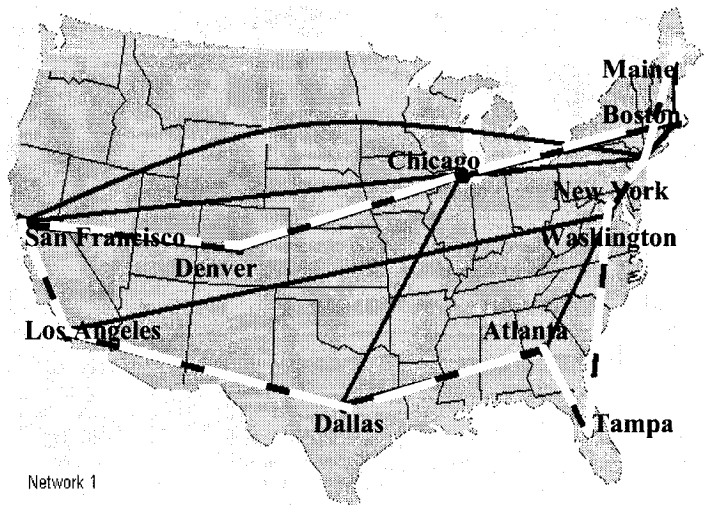


Figure 6.1: Commercial Network Protected By One Cycle

6.2.5.1 Real Network Example

We consider the topology of a commercial network called BBN Planet from the MAPNET database [MAPN05]. The original topology contains some hanging nodes (with only one link connected to the rest of the network) that cannot be protected by any cycle-based link protection and must be removed. The network includes two links with a capacity of one OC-3 each, two links with a capacity of two OC-3, thirteen spans with a capacity of one OC-12 each, and three spans with a capacity two OC-12 each. Since the resulting network is not Hamiltonian, we replace the two OC-12 links (one between Dallas and Kansas City, and the other between Kansas City and Chicago) by a single direct OC-12 link from Dallas to Chicago. The resulting network after aggregation consists of one link with a capacity of OC-3, one link with one OC-6, thirteen spans with one OC-12 each, and three spans with one OC-24. As we can see from Figure 6.1, the network has 11 nodes and 18 links with link capacities ranging from OC-3 to OC-12. It can also be covered by one Hamiltonian cycle shown in black-and-white links.

Table 6-1: Commercial Network Results for Hamiltonian Cycle Restorability

	One cycle	
	Calculated from capacities	Analytical formula (6.13)
Restorability	0.3792	0.4143

The restorability in the first column was calculated from the data provided by the network and using the shown Hamiltonian cycle in the figure, while the value of the second column is obtained from (6.13) for the Shorter Backup Path model.

6.2.5.2 Verification

We have also run simulations to verify our analytical results from (6.12), (6.13) and (6.22). We have generated many random biconnected graphs using [LEDA06] libraries, and then use a program to select 500 networks that can be covered by one Hamiltonian cycle. Restorability is computed for each of the 500 networks for a given F (the max-to-min link capacity ratio). Figure 6.2 shows the mean network restorability performance for the 500 networks as a function of F (with minimum set to 1), each with N=10 nodes and M=18 edges. One can see the analytical result (dashed curve) are very close to the simulation results (small stars with the continues curve). Similar observations are made for the Random-Path case and for the two paths case.

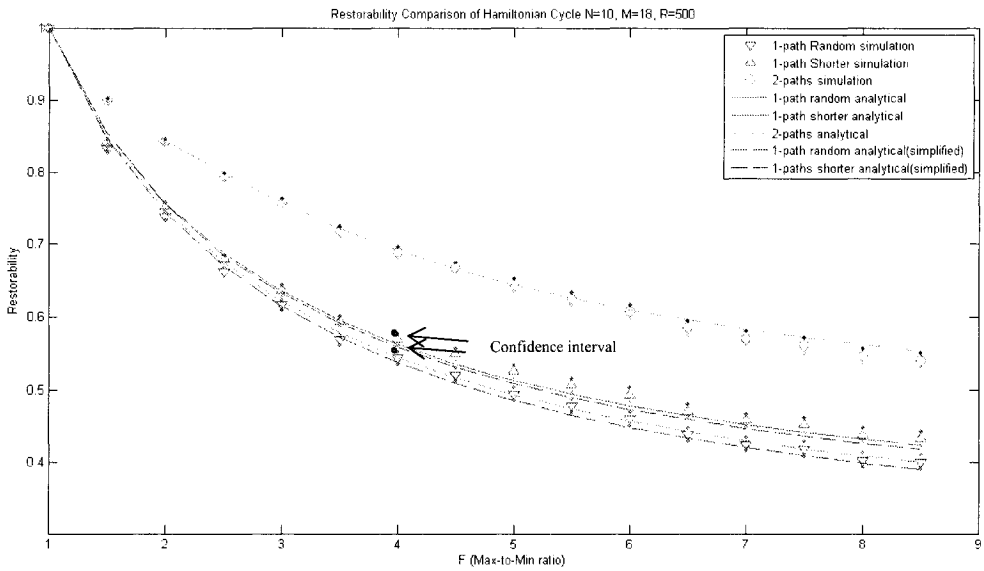


Figure 6.2: Verification of Restorability Analysis

We have also provided the 95% confidence intervals in Figure 6.2. The intervals are very small as indicated by the vertical bars. Since the confidence intervals here and other from other simulations are in general very small, they are omitted in other figures in future.

6.2.6 Performance Evaluation

With our analytical equations verified, we can now go to investigate more properties of protection coverage by one Hamiltonian cycle by studying the restorability performance under various parameters for the three models discussed in Section 6.1 which are now discussed under the One Backup Path (both the Random Backup Path model and the Shortest Backup Path Model) and the Two Backup Path.

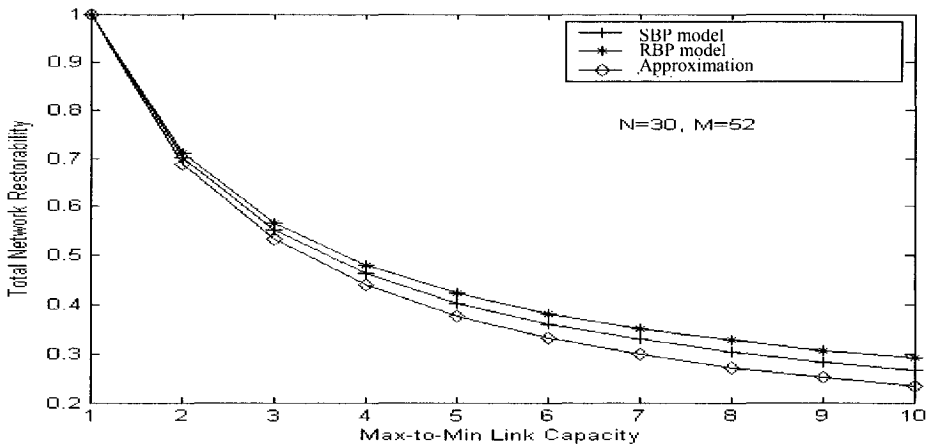


Figure 6.3: Total Restorability vs. Network Capacity Factor

6.2.6.1 One Backup Path

Figure 6.3 presents the restorability (R) performance comparison among the RBP, the SBP and the approximation formula introduced in the previous sections. Looking at the performance curve of the RBP, one sees that R drops sharply as F increases but no as much beyond F=8. The SBP has almost the same performance for small F but its advantages shows up more clearly as F increases. Note that beyond F = 3, R is less than 0.6 and we consider this to be very poor performance, and thus not practical to use a single Hamiltonian cycle to protect the whole network (for example, when links with capacities of OC-192 and OC-12 coexist). Finally, our approximation formula produces not just a lower bound but actually not a bad approximation especially for smaller values of F such as $F \leq 3$ for practical purpose.

Figure 6.4 shows the change of network restorability with the total number of nodes N in the network with the average nodal degree fixed at 3.5, For F=2 and when the network size is below 20 nodes, the number of nodes do affect the restorability negatively and the difference between the random, shorter and the approximation formulas are not significant. However,

for larger values this impact is not noticeable. For F=10, the number of nodes impact on the restorability is noticeable for network size up to around 35 nodes, after which the three curves have a smaller different and have constant behavior. In all cases, we see that the SBP performs better than the RBP. This can be explained by the fact that a longer the path is more likely to have small backup capacity as obtained in (6.6).

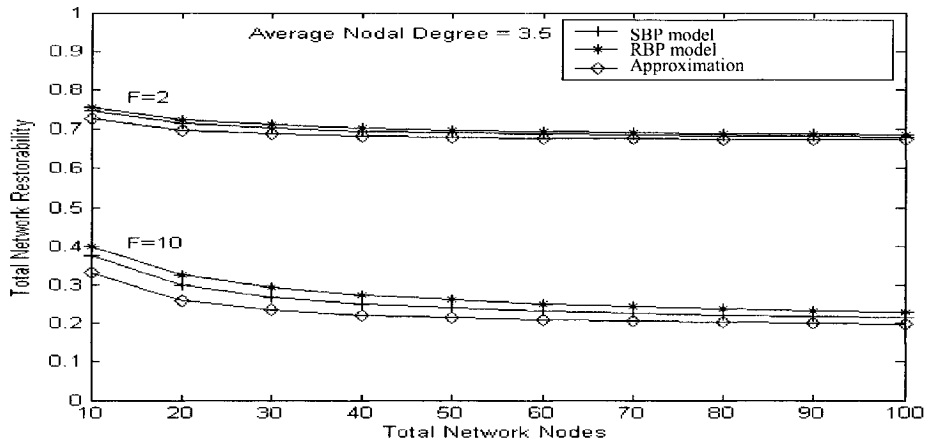


Figure 6.4: Restorability vs. Network Size

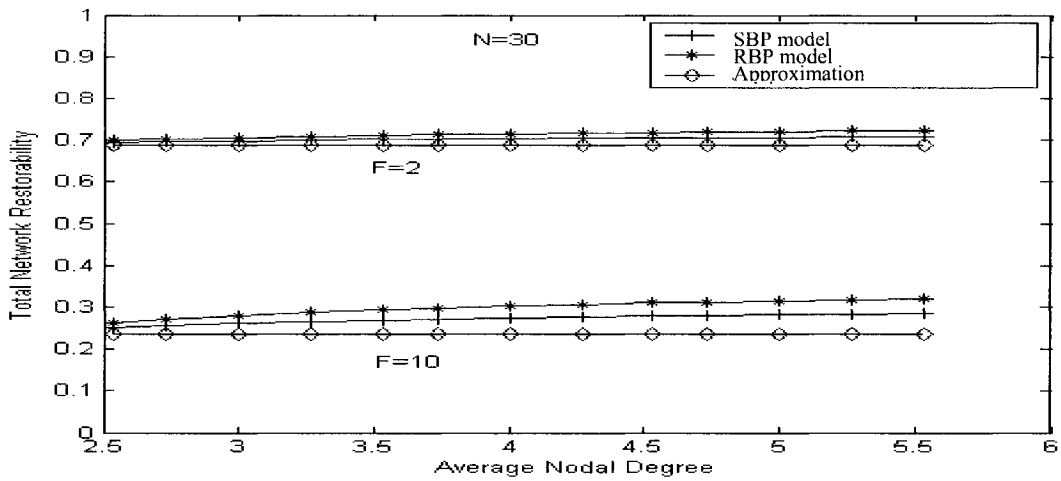


Figure 6.5: Restorability vs. Nodal Degree

Finally, Figure 6.5 shows the performance of network restorability with respect to the average nodal degree for networks with N=30 nodes. The curves are given for two cases of capacity ratios, F=2 and F=10. For F=2, increasing the network degree has little effect on the network restorability with a slight better value for the shortest path. However, for a larger F value (e.g. F=10) the increase in the network restorability is more obvious when we increase

the network nodal degree.

Notice that the difference of approximation formula from the other two models in both Figure 6.4 and Figure 6.5 due to the fact that the approximation formula ignore the increase in the number of edges and its effect in the restorability.

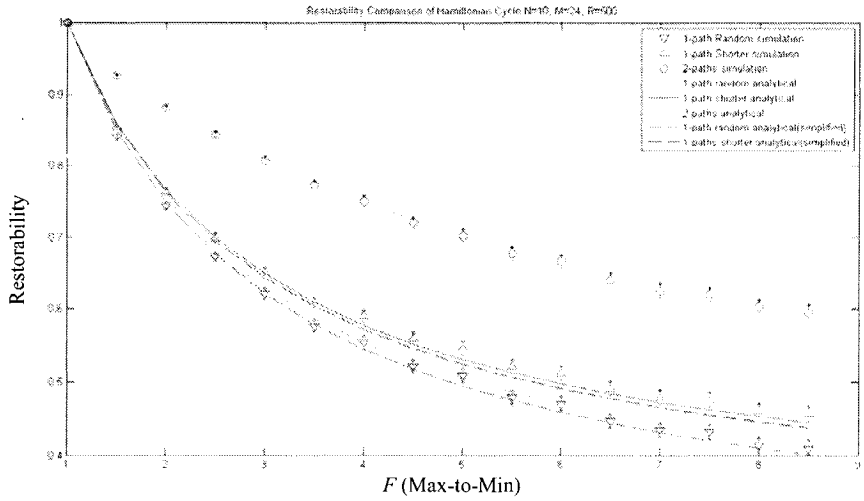


Figure 6.6: Total Restorability vs. Network Capacity Factor Using One and Two Backup Paths

6.2.6.2 Two Backup Paths

Figure 6.6 shows the network restorability comparison between using one backup path and two backup paths for networks with N=10 M=24. The restorability performance of SBP decreases rapidly in the beginning (from R=1 at F=1 to R~0.65 at F=3) before slowing down. The RBP has a close but better performance. On the other hand, the TBP has much better performance than both (e.g. R~0.7 at F=6 compared with R~0.5 for both SBP and RBP)

Additional information/observations can be obtained from Figure 6.2 for a network with N=10 nodes and M=18 links. From both Figure 6.2 and Figure 6.6, increasing the number of edges will increase the restorability of the Hamiltonian cycle. From further comparison, TBP will benefit much more from increasing the number of edges in the graph more than SBP and RBP.

6.3 Double Domain Restorability Analysis

Since a single Hamiltonian cycle is hard to come by sometimes, it should be of interest to assume and to study scenario where the network can be decomposed into D=2 Hamiltonian

subgraphs. However, as we move into the D=2 domain protection, we run into much more intricate mathematical analysis as there are many combinations of scenarios to consider including new ones. In the following, we shall first present some analysis on decomposable graphs that will lead to the concepts of shared versus non-shared capacity between the two domains edges. Any of the previous methods of SBP, TBP and RBP will be used in our analysis where appropriate/desired.

6.3.1 Analysis of Decomposable Graphs

Decomposable graphs are graphs that can have a clear edge cut edge cut defined before in Chapter Two. We are interested in analyzing graphs that can be decomposed into D= 2 Hamiltonian cycles in a network graph G here. In other words, we consider two subgraphs g_1 and g_2 such that $g_1 \cup g_2$ = G and $g_1 \cap g_2 = b$ edges. Let the numbers of nodes and edges in each subgraph be n_1, m_1 and n_2, m_2, respectively. Also assume the intersection between the set of nodes and edges of the two subgraphs is not empty and has at least one edge and two nodes ($b \geq 1$). Thus $n_1+n_2 \geq N+2$ and $m_1+m_2 \geq M+1$ In particular, assume we have exactly b edges and b+1 nodes common between the two subgraphs, then we have $n_1+n_2 = N+b+1$ and $m_1+m_2 = M+b$ (e.g. Figure 6.7). Let $E1$ and $E2$ be the edge set of g_1 and g_2, respectively. Then $E=E1 \cup E2$ *and* $E1 \cap E2 = \{e_1, e_2 \ldots, e_b\}$

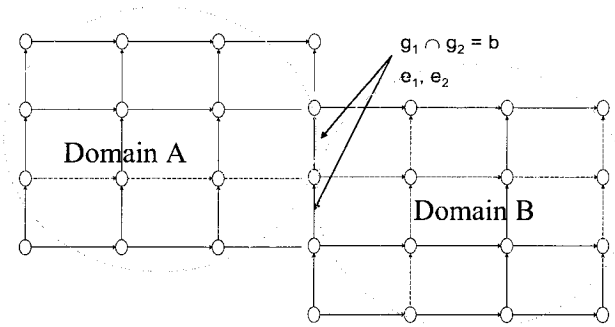


Figure 6.7: Graph Decomposition

The restorability R of the whole graph G and the restorability (R_1 and R_2) of each subgraph are then respectively given by:

$$R = \frac{\sum_{i \in E} c_{wpi}}{\sum_{i \in E} c_{wi}} \qquad (6.23a)$$

$$R_1 = \frac{\sum_{i \in E_1} c_{wpi}}{\sum_{i \in E_1} c_{wi}} \tag{6.23b}$$

$$R_2 = \frac{\sum_{i \in E_2} c_{wpi}}{\sum_{i \in E_2} c_{wi}} \tag{6.23c}$$

To find the sum of protection capacities in G, we need to consider two capacity cases at the intersection between the two cycles. That is, whether the protection capacity in the links of intersection is shared by the two cycles or not, as discussed in the following two sub-sections.

6.3.2 Shared Protection Capacity

In this case where sharing protection capacity between the two cycles is allowed, we should add the additional protection capacity contributed by one cycle on all links of the second cycle except for the amount that is greater than the sum of the protection capacities from both cycles in the shared links. This is captured in (6.24a) below.

$$\sum_{i \in E} c_{wpi} = \sum_{i \in E_1} c_{wpi} + \sum_{i \in E_2} c_{wpi} - \sum_{i=1}^{b} \max(0, c_{wp_1 e_i} + c_{wp_2 e_i} - c_{wp_i}) \tag{6.24a}$$

The sum of working capacities in G is then the total working capacities in subgraph 1 and subgraph 2 edges minus the working capacity at the shared links (since it has been counted twice):

$$\sum_{i \in E} c_{wi} = \sum_{i \in E_1} c_{wi} + \sum_{i \in E_2} c_{wi} - \sum_{i=1}^{b} c_{wi}$$

Observe that R_1 (and similarly R_2) can be written as assuming that the distribution of the capacities in the whole graph is the same as in the two domains

$$R_1 = \left[\frac{\sum_{i \in E_1} c_{wpi}}{m_1}\right] \bullet \left[\frac{\sum_{i \in E_1} c_{wi}}{m_1}\right]^{-1} = \left[\frac{\sum_{i \in E_1} c_{wpi}}{m_1}\right] .(C_{wavg})^{-1} = \frac{\sum_{i \in E_1} c_{wpi}}{m_1 C_{wavg}}$$

Rearranging, we can obtain

$$\sum_{i \in E_1} c_{wpi} = R_1 . m_1 . C_{wavg} \tag{6.25}$$

Note that the above equation assumes the distribution of the capacities in the whole graph is the same as in the two domains including the max-to-min capacity. Hence, F must be the same for each domain and similar to the whole graph.

$$R_1 = \left[\frac{\sum_{i \in E_1} c_{wpi}}{m_1} \right] \bullet \left[\frac{\sum_{i \in E_1} c_{wi}}{m_1} \right]^{-1} = \left[\frac{\sum_{i \in E_1} c_{wpi}}{m_1} \right] . (C_{wavg})^{-1} = \frac{\sum_{i \in E_1} c_{wpi}}{m_1 C_{wavg}}$$

The total network restorability can now be derived using (6.23)-(6.25) as follows:

$$R = \frac{\sum_{i \in E} c_{wpi}}{\sum_{i \in E} c_{wi}} = \frac{\sum_{i \in E_1} c_{wpi} + \sum_{i \in E_2} c_{wpi} - \sum_{i=1}^{b} \max(0, c_{wp_1 e_i} + c_{wp_2 e_i} - c_{wp_i})}{\sum_{i \in E_1} c_{wi} + \sum_{i \in E_2} c_{wi} - \sum_{i=1}^{b} c_{we_i}}$$

$$= \frac{R_1 . m_1 . c_{wavg} + R_2 . m_2 . c_{wavg} - \sum_{i=1}^{b} \max(0, c_{wp_1 e_i} + c_{wp_2 e_i} - c_{wp_i})}{m_1 . c_{wavg} + m_2 . c_{wavg} - \sum_{i=1}^{b} c_{we_i}}$$

$$R = \left[R_1 . m_1 + R_2 . m_2 - \frac{\sum_{i=1}^{b} \max(0, c_{wp_1 e_i} + c_{wp_2 e_i} - c_{wp_i})}{c_{wavg}} \right] . \left[m_1 + m_2 - \frac{\sum_{i=1}^{b} c_{we_i}}{c_{wavg}} \right]^{-1} \tag{6.26a}$$

Finally, using the fact that the capacities are uniformly distributed, we can estimate the average value for R by estimating the average value for the working and the protection capacities, and assuming they basically have the same average value statistically. Then we have

$$\sum_{i=1}^{b} \max(0, c_{wp_1e_i} + c_{wp_2e_i} - c_{we_i}) \approx b.\max(0, R_1 c_{wavg} + R_2 c_{wavg} - c_{wavg}) \tag{6.27}$$

$$\sum_{i=1}^{b} c_{we_i} \approx b c_{wavg} \tag{6.28}$$

Substituting (6.27) and (6.28) in (6.26a), we obtain $\hat{R}$, the average value estimation for R for the shared capacity case as follows.

$$\hat{R} = \frac{R_1.m_1 + R_2.m_2 - \dfrac{b.\max(0, R_1.c_{wavg} + R_2.c_{wavg} - c_{wavg})}{c_{wavg}}}{m_1 + m_2 - \dfrac{b.c_{wavg}}{c_{wavg}}}$$

$$\hat{R} = \frac{R_1.m_1 + R_2.m_2 - b.\max(0, R_1 + R_2 - 1)}{m_1 + m_2 - b}$$

Using the well known relation *2.M=N.d* from graph theory, we can make our estimate explicit in terms of the number of nodes and the nodal degree

$$\hat{R} = \frac{R_1 n_1 d + R_2 n_2 d - 2b\max(0, R_1 + R_2 - 1)}{n_1 d + n_2 d - 2b} \tag{6.29}$$

6.3.3 Not Sharing Protection Capacity

We can follow a development parallel to Section 6.3.2 as follows. Same notation and explanation apply except where noted. Equations (6.24a) and (6.26a) instead will be as shown in (6.24b) and (6.26b). That is, for the protection capacity we have

$$\sum_{i \in E} c_{wpi} = \sum_{i \in E_1} c_{wpi} + \sum_{i \in E_2} c_{wpi} \tag{6.24b}$$

For the restorability, we then have

$$R = [R_1 m_1 + R_2 m_2].\left[m_1 + m_2 - \left[\sum_{i=1}^{b} c_{w_i} \right] / c_{wavg} \right]^{-1} \quad (6.26b)$$

Then, by substituting (6.27) and (6.28) in (6.26b) we obtain the estimate for the R in the non-shared capacity case, i.e.

$$\hat{R} = \frac{R_1 m_1 + R_2 m_2}{m_1 + m_2 - b}$$

Again, by using the number of nodes and the node degree instead of the number of edges we obtain

$$\hat{R} = \frac{R_1 n_1 d + R_2 n_2 d}{n_1 d + n_2 d - 2b} \quad (6.30)$$

As noted in the beginning of the section we can use this formula on the RBP, the SBP or the TBP models as well as for both R_1 and R_2. Note that the formulas found for the restorability of two domains needs the values of R_1 and R_2 as a two sub domains which can be calculated from the single domain formulas generated in Section 6.2.

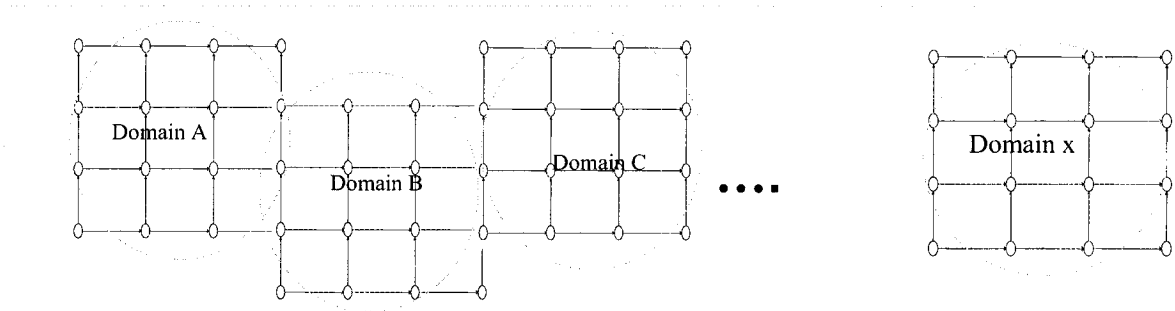


Figure 6.8: Multiple domains interconnections

6.3.4 D-Cycles Extension (D>2)

When we have more than two domains, the formulas above can be applied recursively. The following example will illustrate the idea. Consider a network consists of three domains. Assume that the restorability of the three domains A, B and C are R_A, R_B and R_C, respectively. Then the restorability of the two domains A and B assuming the capacity is not shared between the domains is

$$R_{A,B} = \frac{R_A m_A + R_B m_B}{m_A + m_B - b_1}$$

where m_A and m_B are the number of edges in domain A and B respectivly

Then the restorability of the three domains is the restorability for $R_{A,B}$ and R_C as two domains

$$R_{A,B,C} = \frac{R_{A,B} m_{A,B} + R_C m_C}{m_{A,B} + m_C - b_2}$$

Then

$$\begin{aligned} R_{A,B,C} &= \frac{\frac{R_A m_A + R_B m_B}{m_A + m_B - b_1}(m_A + m_B - b_1) + R_C m_C}{m_A + m_B - b_1 + m_C - b_2} \\ &= \frac{R_A m_A + R_B m_B + R_C m_C}{m_A + m_B + m_C - (b_1 + b_2)} \end{aligned} \tag{6.31}$$

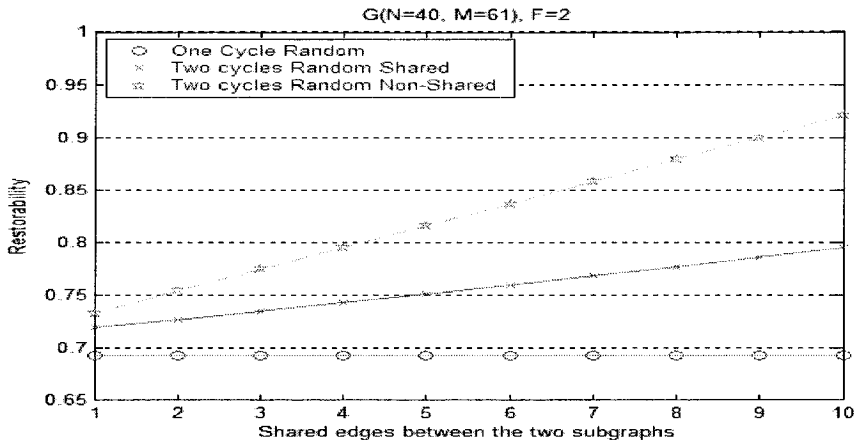
Similar argument can be applied for more domains. In general, we can have:

$$R_{A,B,\ldots,X} = \frac{R_A m_A + R_B m_B + \ldots + R_X m_X}{m_A + m_B + \ldots + m_X - (b_1 + b_2 + \ldots + b_{D-1})} \tag{6.32}$$

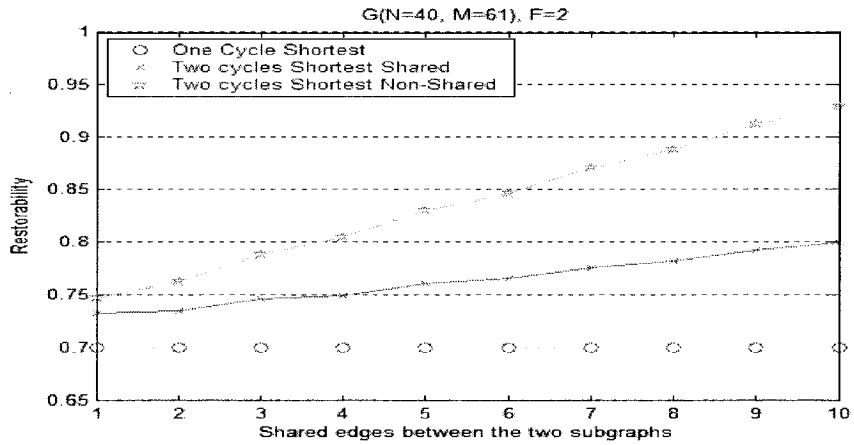
6.3.5 Performance Evaluation

We study in the following the network restorability performance where two cycles are sharing the capacities or not; we also compare it to the one cycle case. Three performance curves are shown in each diagram in this section: one cycle using the RBP (shown as "One Cycle Random"), two cycles with shared protection capacity between the two cycles and using the RBP (shown as "Two Cycles Random Shared"), and the third one is for the non-shared protection capacity using the RBP(shown as "Two Cycles Random Non-Shared"). The experiment is repeated in Figure 6.9b except that we use SBP instead of the RBP.

Figure 6.9a shows the restorability of a graph with respect to the number of shared edges between the two subgraphs. Starting with the "One Cycle Random" curve from the bottom, obviously the network restorability for one cycle is not affected by the shared edges between the two subgraphs and thus the corresponding curve is constant in both figures. It is used here



(a) Using RBP Model



(b) Using the SBP Model

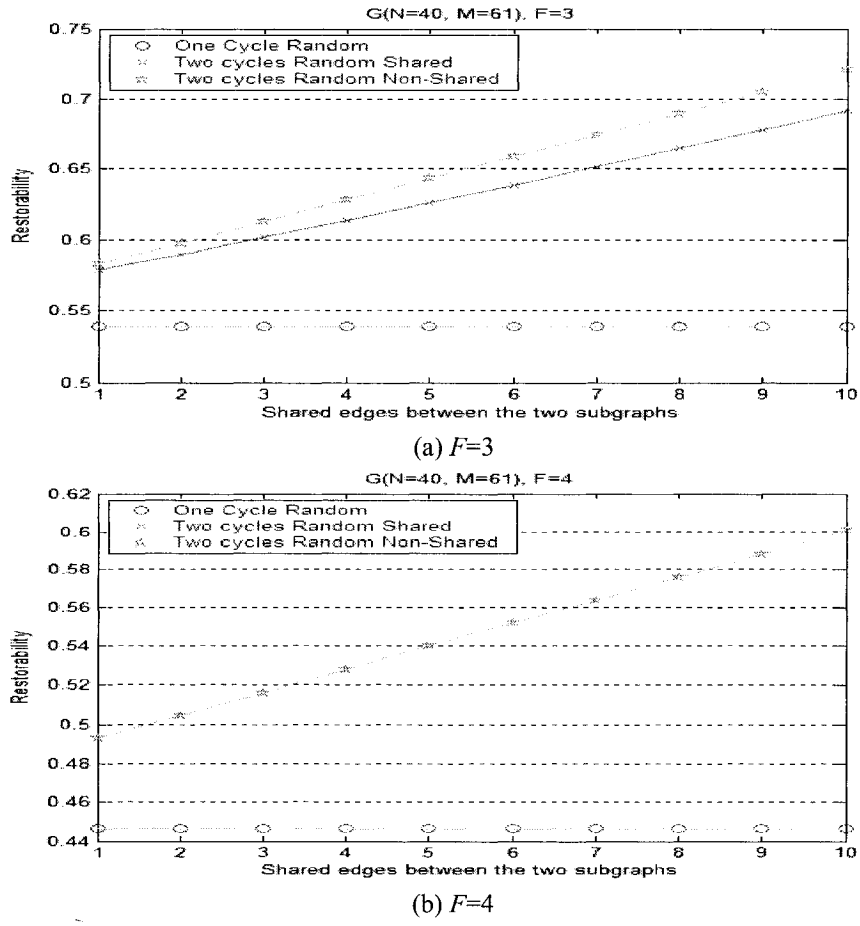
Figure 6.9: Restorability Comparison Using One and Two Equal Cycles for Both the Shared and the Non-Shared Cases. G=(N=40 Nodes, M=61 Edges) and F=2

and in future figures as a reference. As for the Two Cycle Random Shared, the restorability appears to be linearly increasing from 0.72 (with one common edge) to around 0.8 (with 10 common edges). The non-shared case has the best performance as the restorability improves further at the low end (restorability of 0.73 at 1 common edge) and the high end (around 0.92 at 10 common edges). In this figure we assume that the two sub-graphs are equal in size.

For the SBP cases in Figure 6.9b, the observations are very similar, and the results are in fact very close to that of the RBP model.

Figure 6.10 shows the performance for the same network in Figure 6.9b except F is different. Observations similar to Figure 6.9b can be made except the difference between the Two-Cycle Random Shared and Two-Cycle Random non-Shared is smaller in this case where F=3 in Figure 6.10a. In fact for F=4 in Figure 6.10b, there is hardly any difference.

This is expected because the restorability of each sub-graph is a decreasing function of F, and thus both R_1 and R_2 expected to be below 0.5 resulting in $2b\max(0, R_1 + R_2 - 1)=0$ in (6.29). Therefore, the restorability becomes the same as (6.30).

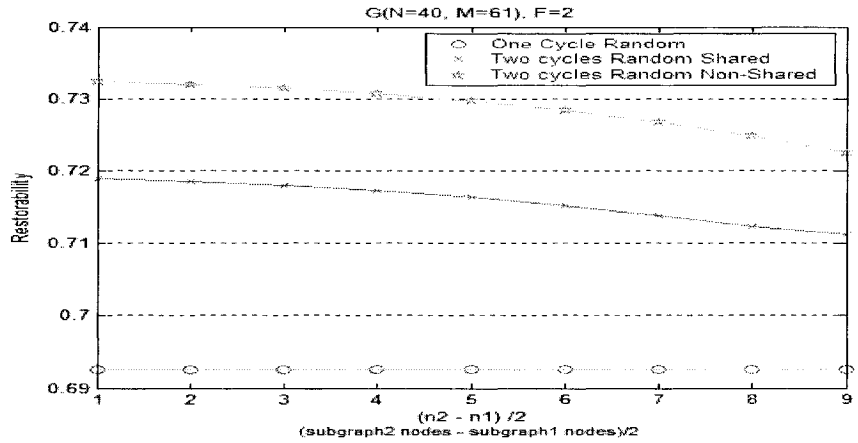


(a) F=3

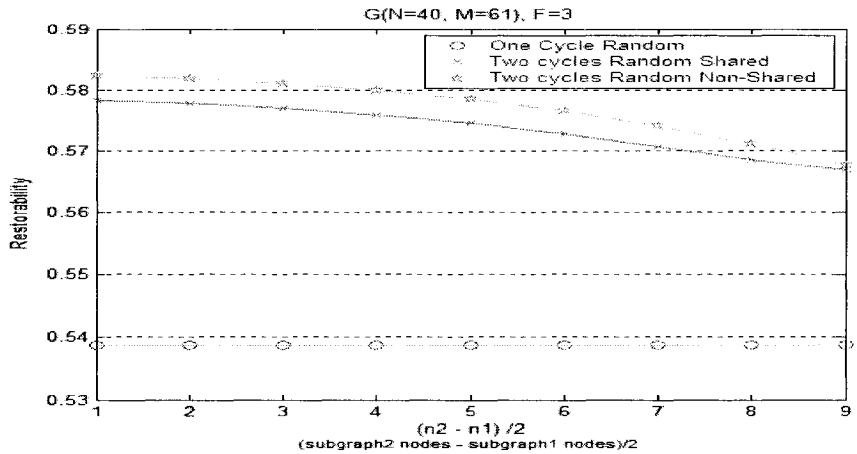
(b) F=4

Figure 6.10: Restorability Comparison for Different F (Max-To-Min Value)

Figure 6.11 studies the effect of the size difference (n2-n1)/2 between the two subgraphs sizes on the restorability (assuming same nodal degree in both subgraphs). In Figure 6.11a, the restorability for the shared case decreases slowly as the difference between the two subgraphs increases. Similar behavior is observed for the non-shared case but with better value. For a higher value of F (Figure 6.11b), we notice similar behavior but a smaller difference between the shared and the non-shared curves.



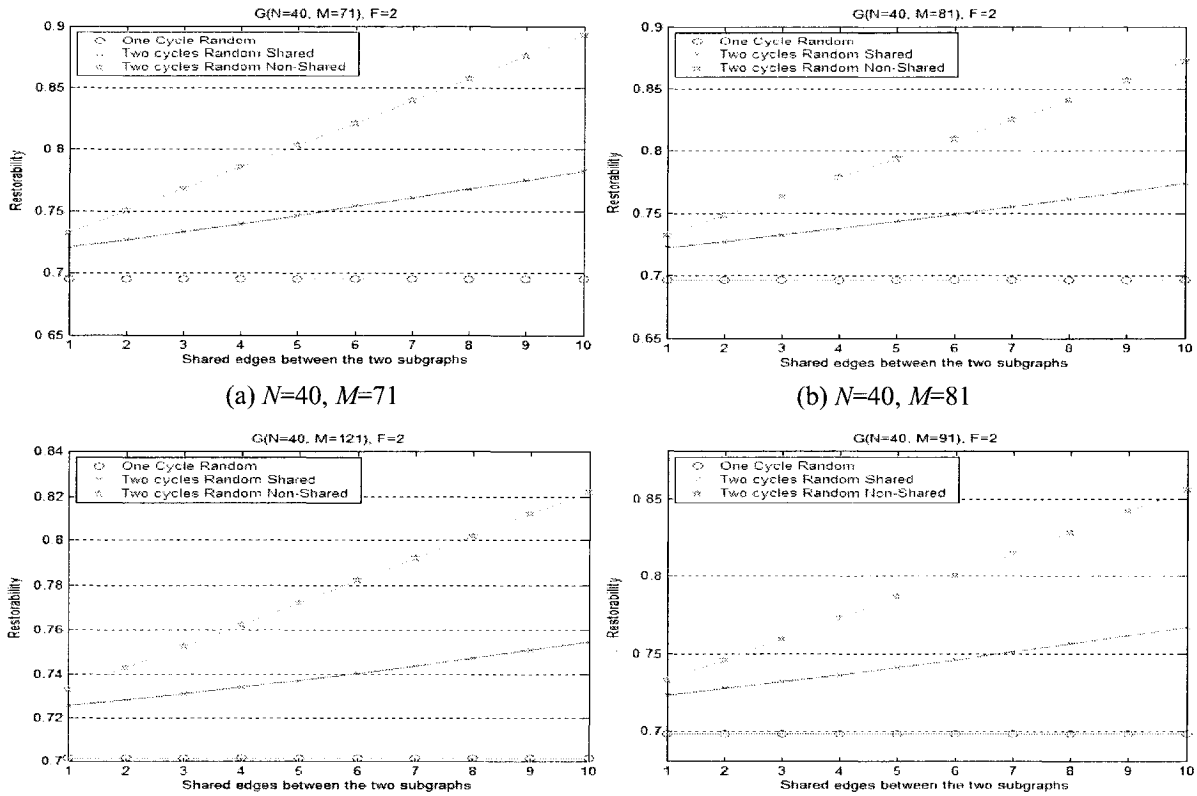
(a) F=2



(b) F=3

Figure 6.11: Restorability Comparison using One and Two Cycles for RBP and for Both the Shared and the Non-Shared Cases. G=(N=40 Nodes, M=61 Edges) and F=2,3

Before we conclude with some results on using more than two cycles, we study the effect of the node degree (2M/N) on the network restorability using two cycles. For the non-shared case we see that the improvement in low-nodal degree network (Figure 6.12a) increases around 0.17 values when moving from 1 to 10 shared edges, whereas as the nodal degree increases (Figure 6.12d) the improvement is around 0.09 values, which means that in the dense networks the improvement in the restorability using two cycles is less significant than the sparse networks.



(a) N=40, M=71 (b) N=40, M=81

(c) N=40, M=91 (d) N=40, M=121

Figure 6.12: Restorability Comparison for Different Network Sizes with Same Number of Nodes and Increased Number of Edges

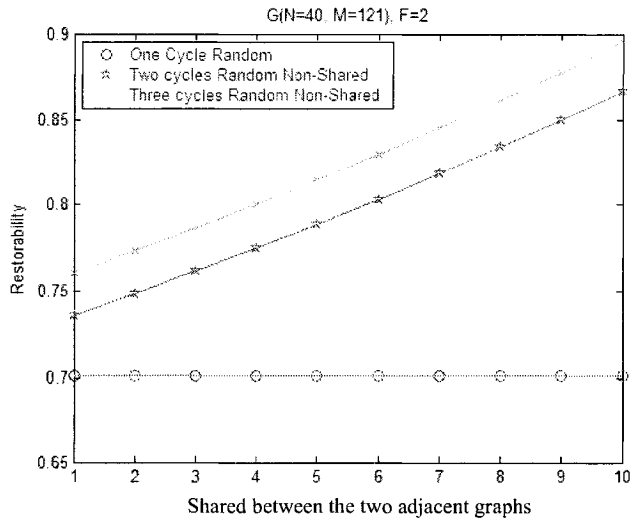


Figure 6.13: Restorability Comparison between Using One, Two and Three Cycles to Protect a Network

Figure 6.13 shows a comparison between the restorability of using two cycles and three

cycles to cover the graph with N= 40, M=121, and F=2. One can see that, like the two-domain case, the restorability is almost a linearly increasing function of the number of edges shared between two adjacent graphs. The improvement, as expected, is around 0.02%. This is due to the availability of more protection capacity for the straddling links on each cycle.

Finally, it is interesting to note that although the restorability formula for two domains assumes the same F values in both domains, it can also be used as a rough approximation for domains with different F values. For example, in the same commercial network of Fig. 6.1, the two cycles we come up with have F=2 and 4 respectively. On using the formula of (6.30), we obtain a restorability of 0.4912, which is not very far away from the figure of 0.4725 obtained based on capacities. The validity of the approximation, however, needs to be further verified by more case studies.

6.4 Network Redundancy Analysis

After we have analyzed the restorability for different cases of single and double domains we study now another performance measure which is the network redundancy. Define *network redundancy* (R_d) to be the ratio of the protection capacity to the working capacity of the network in order to achieve maximum restorability (100%) for a given network.

6.4.1 Single Domain Analysis

We will provide the worst and the best case analysis here as the mean value analysis solution has so far eluded us. The assumption in the analysis is that the whole network can be covered by one Hamiltonian cycle. Define R_{d_worst} to be in the redundancy required in the worst case which occurs when the link with the maximum working capacity (c_{wmax}) is an on-cycle link, because every other link in the cycle must then have a protection capacity equal to c_{wmax} in order to achieve full restorability. This also occurs when the straddling link has the maximum working capacity and only one backup path (shortest or random) is used which again require every other link on the cycle to have capacity of c_{wmax}.

In view of the above, we must assign protection capacities equal to c_{wmax} to every cycle link in order to guarantee 100% restorability. Therefore, the total protection capacity in the network equal to $C_{wmax} \times N$. Define P_r to be the ratio of maximum to average link working capacity and D_{avg} the average nodal degree of the network, we can write

$$R_{d\_worst} = \frac{N \times C_{w\max}}{M \times C_{avg}} = \frac{N \times C_{w\max}}{M \times (\frac{C_{\min} + C_{\max}}{2})} = \frac{4F}{D_{avg}(1+F)} \quad (6.33)$$

The best case scenario occurs when we are using both backup paths for the straddling links and all the links with a higher working capacity are straddling links. i.e. no on-cycle link has working capacity more than any straddling link. The analysis of this scenario is rather hard. So far, we have the results when the number of straddling links is more than the number of on-cycle links (i.e. nodal degree ≥4). Under this case, we have similar to the above except that N is multiplied by half of the C_{wmax} to account for all the on-cycle links that might have less capacity that the straddling links and for the nodal degree which is halved, Then:

$$R_{d\_best} = \frac{N \times \frac{C_{w\max}}{2}}{M \times c_{avg}} = \frac{N \times \frac{C_{w\max}}{2}}{M \times (\frac{C_{\min} + C_{\max}}{2})} = \frac{2F}{D_{avg}(1+F)} \quad (6.34)$$

6.4.2 Two Domains Analysis

If the protection capacity is shared between the two cycles then the worst redundancy value will be a bit better than the single cycle case since at the worst case we assume that the maximum protection capacity exists in each of the two cycle domain. i.e.

$R_{d\_worst\_two\_shared} = R_{d\_worst}$

If the protection capacity is not shared, then we have extra protection capacity equal to the number of shared edges times the maximum capacity value. i.e.

$$R_{d\_worst\_two\_not\_shared} = \frac{N \times C_{w\max} + b.C_{w\max}}{M \times C_{avg}} = \frac{N \times C_{w\max} + b.C_{w\max}}{M \times (\frac{C_{\min} + C_{\max}}{2})} = \frac{(4+b)F}{D_{avg}(1+F)} \quad (6.35)$$

where b is the number of shared edges.

The best case can be obtained when all the high working capacity is all included in one cycle in a similar fashion to the case discussed in the previous subsection. For the other cycle it will contain in the best case a much smaller value than the first one. Again this best case would occur at high nodal degree value ≥4.

$$R_{d\_best} = \frac{n_1 \times \frac{c_{w\max}}{2} + n_2 \times \frac{c_{w\max}}{4}}{M \times c_{avg}} = \frac{\frac{(n_1 + N)c_{w\max}}{4}}{M \times (\frac{C_{\min} + C_{\max}}{2})} \tag{6.36}$$

$$= \frac{(n_1 + N)c_{w\max}}{2M \times (C_{\min} + C_{\max})} = (\frac{n_1}{N} + 1)\frac{F}{D_{avg}(1+F)}$$

6.4.3 Performance Evaluation

In Figure 6.14 we show the relation between the redundancy and the nodal degree for different Max-to-Min ratio. In flat networks (F=1) we see that the highest value starts at nodal degree equals 2 with 100% redundancy drops to around 40% at high nodal degree of 5. We notice similar behavior for the higher value of F where we need 150% redundancy for networks with nodal degree of 2 with F=3 which drops to 60% redundancy for networks with nodal degree of 5. Again we see similar behavior for F=8.

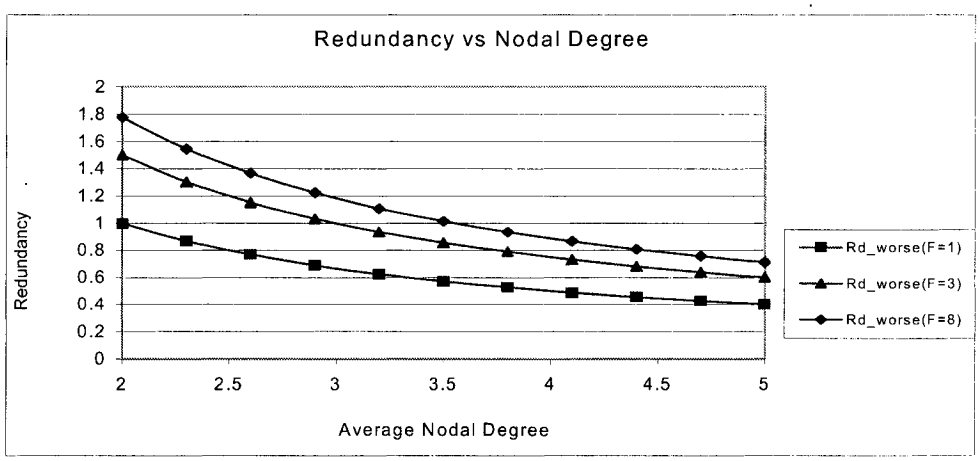
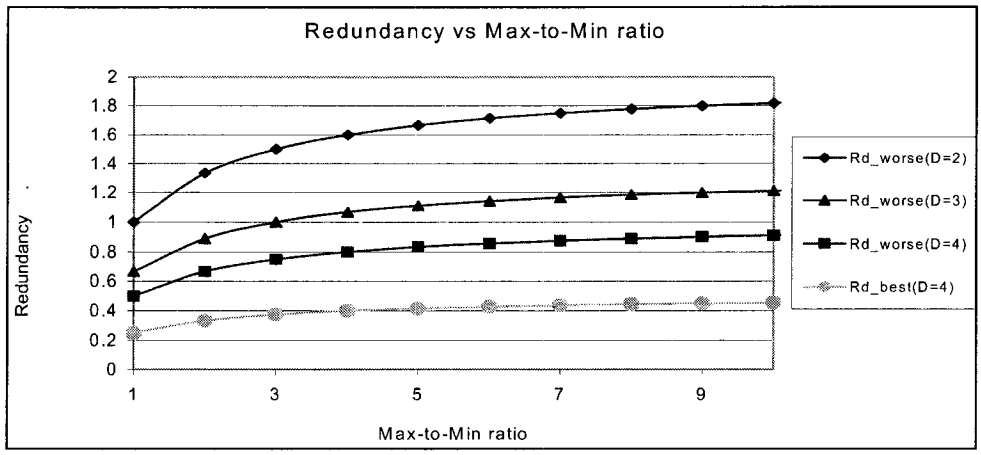
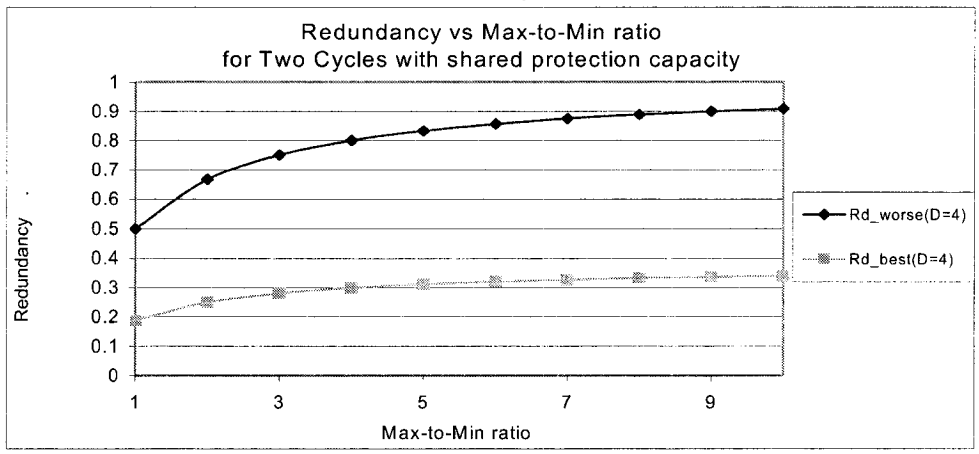


Figure 6.14: Redundancy and Nodal Degree

In Figure 6.15 we compare the redundancy requirement changes with the Max-to-Min ratio for different nodal degree. Figure 6.15a shows the one cycle case and Figure 6.15b shows the two cycles case. In Figure 6.15a, we see that the redundancy required increases for D=2 case from 100% at F=1 to around 180% at F=10, whereas the redundancy increases from around 50% to 90% when moving from F=1 to F=10 with D=4. For the best case scenario for D=4 we see that the restorability changes from around 20% at F=1 to around 35% at F=10.



(a) One Cycle



(b) Two Cycles

Figure 6.15: Redundancy and Max-To-Min Ratio

In Figure 6.15b, we see that the redundancy required for the worst case increases for D=4 case from 50% at F=1 to around 90% at F=10, whereas the best redundancy increases from 20% to 35% for similar values of F.

6.5 Average Protection Path Length Analysis

After we have analyzed the restorability and the redundancy for different cases of single and double domains we study a related performance measure which is the average path length. We will do that for single and two domains as well.

6.5.1 Single Domain

Since there is one cycle covering a network with N nodes and M links, there are N on-cycle links and M-N straddling links. Each of the on-cycle links has a protection path length of N-1 (the portion of the cycle in the opposite direction). As discussed before, each straddling link has a protection path length k within a range of $2 \leq k \leq N$-2 when we use random choice and

within $2 \leq k \leq N/2$ when we use the shorter. If we denote the length of an individual backup path in the network by h_i, $i=1,...,M$, the average backup path length H_x could be analyzed as follows:

$$H_x = \frac{1}{M}\sum_{i=1}^{M} h_i$$

Where x can be RBP, SBP or TBP

Random Backup Path Case

For choosing one of the two alternate paths randomly with an equal probability, the average protection path length seen by a straddling link is $(N+1)/2$ hops. Thus the average backup path length for the network is

$$H_{RBP} = \frac{1}{M}\left(N \times (N-1) + (M-N) \times \frac{N+1}{2} \right) = \frac{1}{2M}\left(N^2 - 3N + M.N + M \right) \quad (6.37)$$

Shortest Backup Path Case

If one chooses the shorter of the two paths, the protection path length could be anywhere between 2 and $N/2$, implies $(N+4)/4$ on the average. Then

$$H_{SBP} = \frac{1}{M}\left(N \times (N-1) + (M-N) \times \frac{N+4}{4} \right) = \frac{1}{4M}\left(3N^2 - 7N + M.N + 4M \right) \quad (6.38)$$

Two Backup Paths Case

When both backup paths are used, the total length of the two paths is always N. The protection path length for the on-cycle link is always N-1. Then the average protection path length for any link

$$H_{TBP} = \frac{1}{M}\left(N \times (N-1) + (M-N) \times N \right) = \frac{-N}{M} + N = N - \frac{2}{D_{avg}} \quad (6.39)$$

6.5.2 Two Domains

When using two cycles for protection the average protection path will be reduced down to half of all the values found in the previous section when the two cycles are almost equal in size. This is intuitive because the recovery path length for an on-cycle link will need to

traverse $N/2$-1 links rather than N-1 links. For the straddling links, the range for the straddling links recover path length is 2 to $N/2$ rather than 2 to N-2. The mean path length for the three backup path models are then the following

$$H_{RBP\_two} = \frac{1}{4M}\left(N^2 - 3N + M.N + M\right) \tag{6.40}$$

$$H_{SBP\_two} = \frac{1}{8M}\left(3N^2 - 7N + M.N + 4M\right) \tag{6.41}$$

$$H_{TBP\_two} = \frac{N}{2} - \frac{1}{D_{avg}} \tag{6.42}$$

When the two cycles are different in size the analysis becomes harder since we have to find the probability of each backup path length of each link relative to its protection cycle size.

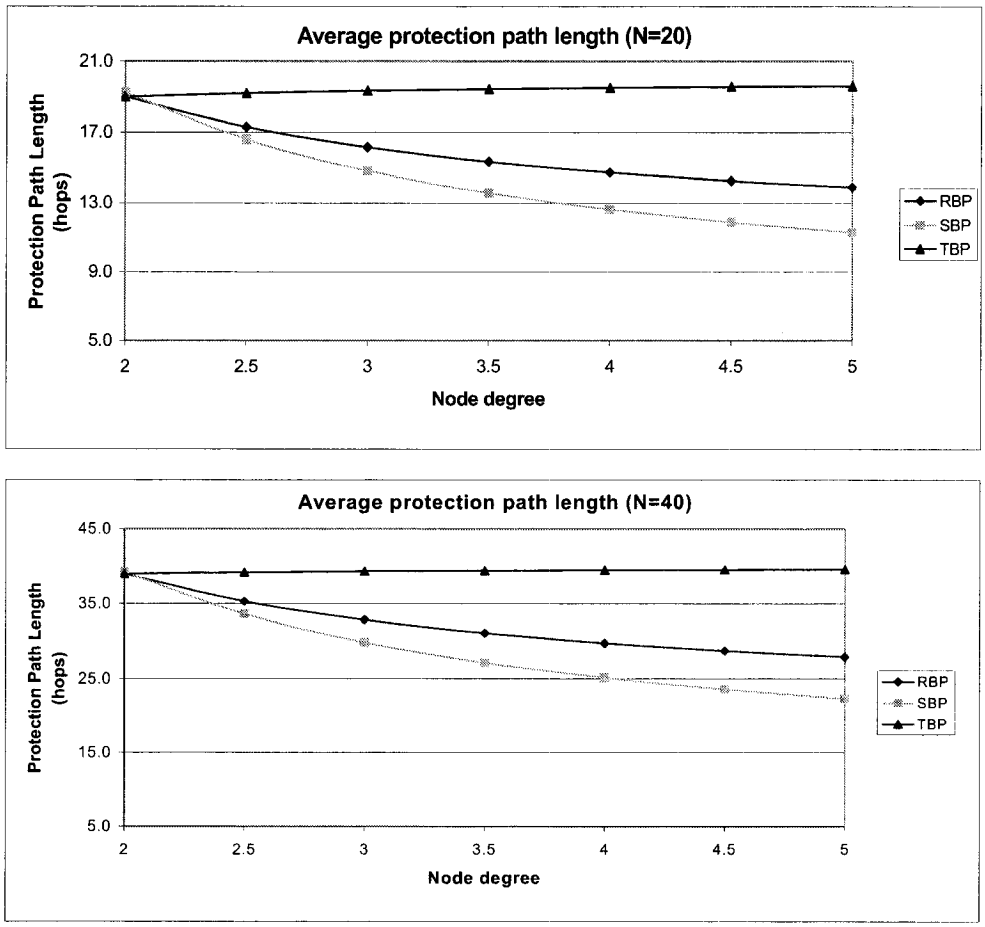


Figure 6.16: Average Protection Path Length Comparison for N=20 and 40 Nodes and Different Nodal Degree

6.5.3 Comparison

A comparison between the average protection path lengths for a single cycle is shown in Figure 6.16. Figure 6.16a shows the comparison for a N=20 nodes network and Figure 6.16b for N= 40 nodes network. Under the TBP model in Figure 6.16a, the average protection path length increases very slowly from 19 at nodal degree of 2 to 19.6 at a nodal degree of 5. However, when using a single backup protection path we have different performance. Under the RBP model, the path length decreases from 19 to around 14 when the nodal degree is increased from 2 to 5. The SBP can improve further in that the protection path decreases from 19 down to around 11. We observe similar behavior for larger network s (N=40 nodes) in Figure 6.16b.

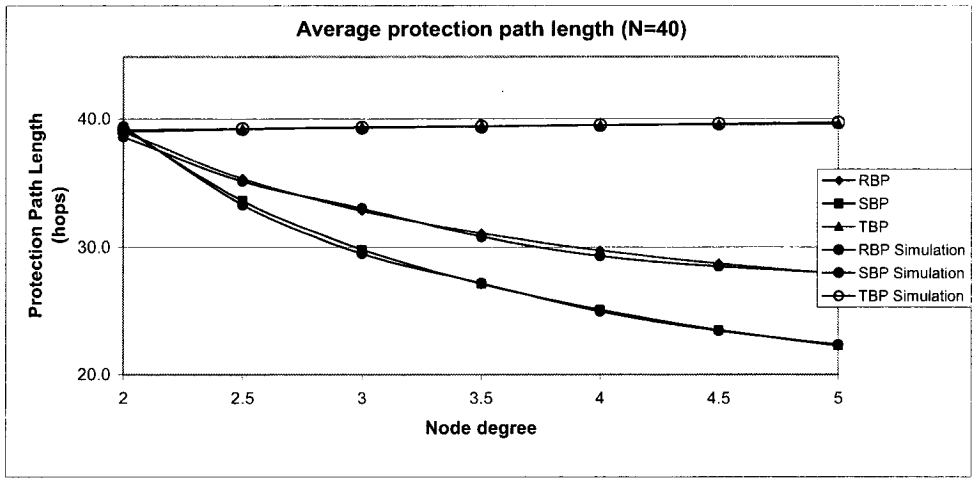


Figure 6.17: Comparison Between the Analytical and the Simulation Results for the Average Protection Path Length for N=20 and 40 Nodes and Different Nodal Degree

Finally, we present in Fig. 6.17 a comparison between the analytical and simulation results obtained from the random graphs used in the previous sections. We can see a very close match between the two results.

6.6 Concluding remarks

In this chapter, we have presented mathematical analysis of network restorability, redundancy, and average protection path in mesh networks using one domain (protection cycles). We have also looked the two domain model that has been suggested in the literature. Even though we have obtained limited results for the Two-Domain Protection, we nevertheless shed some light into the behavior of the network. The extension we provide in the decomposition (two cycles) could also be applied to any number of protection cycles by

applying the derived results for the intersected pairs of cycles. More work would employ analyzing more general cases where the intersection between the two subgraphs (Hamiltonian cycles) is more arbitrary than our simplified assumption.

Our results can be used by network planners to forecast the restorability of a given network, or by network architects to find more efficient techniques for link protection in hierarchical or random mesh networks. In fact the network designers could fine-tune their networks to target the theoretical bounds we provide in this paper. Possible tuning includes, but not limited to, the size of the network domains, the shared edges between domains, and the node degree within each domain.

In general, we feel that using one protection path for a straddling link in very high speed networks is more practical for quick and simpler operation of switching the traffic. In this regard, SBP have been found to have a bit better performance than RBP which suggest using it rather than random choice. When optical technology has progressed to allow the splitting the traffic at some point on the cycle and re-aggregating it at another point, TBP will be a better choice.

Chapter Seven
Conclusions

We have proposed a novel idea to generate cycles from planar graphs that can be applied to mesh networks. Using a chordal graph approach, we have found the sufficient conditions to formulate and to guarantee the existence of PCC (Planar Chordal Cycle) to cover a set of faces in the mesh network. Using the dual graph concept, we have also invented the Modified Cycle Graph (MCG) that can reduce the complexity of identifying and constructing PCCs.

By including a set of searching techniques and criteria, we are able to come up with algorithms to construct cycles with different objectives and properties. In fact, these are put in work in the applications of multicasting and network protection, two research areas that receive much interest nowadays, Performance evaluations have been carried out for these two applications which in turn help to test out the performance and properties of the PCC in addition to demonstrating the capability of our algorithms to generate the cycles very fast (in polynomial time). Of course, nothing is perfect. Efficiency and performance study have been conducted to compare our MCG technique to other techniques, and some tradeoff study performed.

We have also analyzed some performance measures for network cycle-based protection mechanisms. We have presented detailed study for restorability analysis for single protection cycle which is useful for single domain networks, and we have also presented redundancy and protection path analysis. For two domains network that would use single cycle for each domain, we have presented analytical model that uses the single domain idea and considering the shared edges between the domains. Our work provides some foundation for future analysis of multi-domain protection.

7.1 Future Work

Based on this thesis work we suggest the following future work:

- Extending the idea of PCC for non-planar graph using the general cycle basis idea rather than the planar cycle basis of planar graphs.

- Implement signaling protocols for protected multicasting using PCC.
- Extending the analysis and performance study in Chapter Six for more than two cycles. Also, to extend the analysis for other distributions rather than just uniform random distribution.
- More work would employ analyzing more general cases where the intersection between the two subgraphs (Hamiltonian cycles) is more arbitrary than our simplified assumption.
- Taking into account the bandwidth requirement in multicasting.
- Comparing the presented cycle-based approach and similar approaches with non-cycle based approaches such as the tree-based methods.

References

[AAPN03] Agile All-Photonic Networks, http://www.aapn.mcgill.ca/

[AdSa05] Adil Kodiana, Anthony Sackb, Wayne D. Grovera, "The threshold hop-limit effect in p-cycles: Comparing hop- and circumference-limited design," *J. of Optical Switching and Networking*, Vol. 2, No. 2, 2005. pp. 72–85.

[AMPL06] AMPL v. 7.0, http: //www.ampl.com/ as of 2006.

[ArGr00] P. Arijs, M. Gryseels, and P. Demeester, "Planning of WDM Ring Networks," *J. of Photonic Network Communications*, Vol. 2, No. 1, 2000, pp. 33-51

[AsSh01] C. Assi, A. Shami, M. A. Ali, R. Kurtz, D. Guo, "Optical networking and real-time provisioning: an integrated vision for the next-generation Internet," *IEEE Network Magazine*, July/August 2001, pp.36-45.

[AtLe89] H. Attiya, J. van Leewen, N. Santoro, S. Zaks. "Efficient elections in chordal ring networks," *Algorithmica*, Vol. 4, 1989, pp. 437-446.

[BoMu82] J. A. Bondy and U. S. R. Murty. "Graph Theory with Applications," North Holland, New York, Amsterdam, Oxford 1982

[CaGl66] D. Cartwright and T. C. Gleason, "The number of paths and cycles in a digraph," *Psychometrika*, Vol. 31, 1966, pp. 179-199.

[ChEl00] G.K. Chang, G. Ellinas, B. Meagher, W. Xin, S.J. Yoo, M.Z. Iqbal, J. Young, H. Dai, Y.J. Chen, C. Lee, X. Yang, A. Chowdhury, T.F. Chen, "a proof-of-concept ultra-low latency optical label switching testbed demonstration for next generation Internet networks," *Optical Fiber Communication Conference* (OFC 2000), Vol. , 2000, pp. 56–58.

[ChNi89] N. Chiba, T. Nishizeki, "The Hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs," *J. of Algorithms*, Vol. 10, No. 2, June 1989), pp. 187 – 211.

[Cisc04] Cisco 2004, "Fundamentals of DWDM Technology," http://www.cisco.com/univercd/cc/td/doc/product/mels/cm1500/dwdm/dwdm_ovr.htm

[CoLi02] G. Conte, M. Listanti, M. Settembre, R. Sabella, "Strategy for Protection and Restoration of Optical Paths in WDM Backbone Networks for Next-Generation Internet Infrastructures," *J. of Lightwave Technology*, Vol. 20, No. 8, Aug. 2002, pp. 1264 -1276

[CPLE06] CPLEX v. 7.0, http: //www.ilog.com as of 2006.

[DeDe98] H.M. Deitel and P.J. Deitel, "C++: How to Program," 2nd ed., Prentice Hall, 1998.

[Deo79] Narsingh Deo, "Minimum-Length Fundamental Cycle Set," *IEEE Transactions on Circuits and Systems*, Vol. Cas-26, No. 10, October 1979.

[DePr82] Narsingh Deo and G. M. Prabhu, "Algorithms for Generating Fundamental Cycles in a Graph," *ACM Transactions on Mathematical Software*, Vol. 8, No. 1, March 1982, pp. 26-42

[DiPo03] Aijun Ding and Gee-Swee Poo, "A survey of optical multicast over WDM networks," *Computer Communications*, Vol. 26, No. 2, Feb. 2003, pp. 193-200.

[DiMa91] D.D. Dimitrijevic, B. Maglaris, R. R. Boorstyn, "Routing in multi-domain networks," *Proceed. of. Tenth Annual Joint Conference of the Networking in the 90s* (INFOCOM '91), 7-11 April 1991, Vol.1, pp. 257-264.

[DoHe03] J. Doucette, Donna He, W. D. Grover, Oliver Yang, "Algorithmic Approaches for Efficient Enumeration of Candidate p-Cycles and Capacitated p-Cycle Network Design," *Proceed. of Workshop on Design of Reliable Communication Networks* (DRCN 2003), Banff, AB, Canada, 19-22 October 2003, pp. 212-220.

[DoKr96] U. Dogrusoz and M. S. Krishnamoorthy, "Enumerating All Cycles of a Planar Graph," *J. of Parallel Algorithms and Applications*, Vol. 10, 1996, pp. 21-36.

[DoLe93] Doar, M., Leslie, I., "How Bad is Naive Multicast Routing," *Proceed. of IEEE INFOCOM'93*, San Francisco, CA, Apr. 1993, pp. 82-89.

[FeYu98] Gang Feng, Tak-Shing Peter Yum, Siew Chee Kheong, "Dynamic Multicasting On Bi-Directional Rings," *Electronics Letters*, 6 July 2000,Vol. 36, Issue 14, pp. 1251-1253.

[FlHu06] Paola Flocchini, Miao Jun Huang, Flaminia L. Luccio, "Decontamination of Chordal Rings and Tori," *Proceed. of 20th International Parallel and Distributed Processing Symposium 2006* (IPDPS 2006). 25-29 April, 2006, pp. 8-16.

[GaHe94] L. M. Gardner, M. Heydari, J. Shah, I.H. Sudborough, I.G. Tollis and C. Xia, "Techniques for Finding Ring Covers in Survivable Networks," *Proceed. of IEEE Global Telecommunications Conference 1994* (GLOBECOM '94.), San Francisco, CA, USA, 28 Nov- 2 Dec 1994, Vol. 3, pp. 1862-1866

[GaJo76] M. R. Garey, D. S. Johnson, R. E. Tarjan, "The planar Hamiltonian circuit problem is NP-complete," *SIAM J. on Computing*. Vol. 5, 1976, pp. 704-714.

[GoLi02] M. Goyal, G. Li, J. Yates, "Shared Mesh Restoration: A Simulation Study," *Optical Fiber Communication Conference 2002* (OFC 2002), 17-22, March 2002, Anaheim, California, pp. 489 - 490

[GrDo02] Grover, W: D. Doucette, J. E., "Advances in Optical Network Design with p-Cycles: Joint Optimization and Pre- selection of Candidate p-Cycles," *Proceed. of the Topical Meeting on All Optical Networking*, Mont Tremblant, Quebec, July 15- 17, 2002, pp. 49-50 (paper WA2).

[Grov02] Wayne Grover, "understanding p-cycles, enhanced rings, and oriented cycle covers," *Proceed. of 1st International Conference on Optical Communications and Networks* (ICOCN '02), Singapore, November 11-14, 2002, pp. 305-308.

[Grov03] Wayne D. Grover, "Mesh-based Survivable Networks," Prentice Hall, 2003.

[Grov98] Wayne D. Grover, "Cycle-Oriented Distributed Preconfiguration: Ring-like Speed with Mesh-like Capacity for Self-planning Network Restoration," *Proceed. of IEEE International Conference on Communications*, Atlanta June 7-11, 1998, pp. 537-543.

[GrSc02] C.G. Gruber and D.A Schupke, "Capacity-efficient Planning of Resilient Networks with p-cycles," *Proceed. of 10th International Telecommuniation Network Strategy and Planning Symposium 2002.*

[GrSh03] Wayne D. Grover and Gangxiang Shen, "Extending the p-cycle concept to path-segment protection," *Proceed. of IEEE International Conference on Communications 2003,* ICC2003, Anchorage, Alaska, USA, May 11-15, 2003, pp. 1314-1319.

[HaMu02] Oded Hauser, Murali Kodialam, T. V. Lakshman, "Capacity Design of Fast Path Restorable Optical Networks," *Proceed. of IEEE INFOCOM 2002*, Vol. 2, pp. 817-826.

[HeFu99] Helmut Alt, Ulrich Fuchs and Klaus Kriegel, "On the Number of Simple Cycles in Planar Graphs," *Probability and Computing Combinatorics*, Vol. 8, No. 5, pp. 397 – 405, September 1999.

[HeBy95] M. Herzberg, S.J. Bye, A. Utano, "The hop-limit approach for spare-capacity assignment in survivable networks," *IEEE/ACM Transactions on Networking*, Vol. 3, No. 6, 1995, pp. 775–784.

[Hort87] J. D. Horton, "A Polynomial-Time Algorithm to find the shortest cycle basis of a graph," *SIAM J. on computing,* Vol. 18, No. 2, April 1987.

[HoTa74] John Hopcroft and Robert Tarjan, "Efficient Planarity Testing," *J. of the ACM*, Vol. 21, No. 4, pp. 549-568, 1974,

[HsHo72] H. T. Hsu and P. A. Honkanen, "A Fast Minimal Storage Algorithm for Determining All the Elementary Cycles of A Graph," Computer Science Dept., Pennsylvania State Univ., University Park, 1972

[HuCo01] Hong Huang and Johan Copeland, "Hamiltonian Cycle Protection: a Novel Approach to Mesh WDM Optical Network Protection," *Proceed. of IEEE Workshop on High Performance Switching and Routing*, 29-31 May 2001, pp. 31-35.

[HuCo02] Huang, J. A. Copeland, "Multi-domain mesh optical network protection using Hamiltonian cycles," *Proceed. of IEEE Workshop on High Performance Switching and Routing 2002* (HPSR 2002), Kobe, Hyogo, Japan, 26-29 May 2002, pp. 83-87.

[IEC02] International Engineering Consortium, "Optical Networks " http: //www.iec.org/online/tutorials/opt_net/topic10.html?Back.x=8&Back.y=17

[ITU99] ITU-T Recommendation G.872. Architecture of Optical Transport Networks (OTN), 1999.

[JaDh01] Raj Jain and Sudheer Dharanikota, "Internet Protocol over DWDM - Recent Developments, Trends and Issues," Global Optical Communications - Business Briefing published by World Market Research Centre Ltd (www.wmrc.com), London, UK, 10, July 2001.

[JeXi00] M. Jeong, Y. Xiong, H.C. Cankaya, M. Vandenhoute, C. Qiao, "Efficient multicast schemes for optical burst-switched WDM networks," *IEEE International Conference on Communications 2002* (ICC 2000), June 18-22, 2000, New Orleans, USA, pp. 1289-1294.

[JiDu01] X.-H. Jia, D.-Z. Du, X.-D. Hu, M.-K. Lee, J. Gu, "Optimization of wavelength assignment for QoS multicast in WDM networks," *IEEE Transactions on Communications*, Vol. 49, No. 2, 2001, pp. 341–350.

[JiZh01a] Xiaohua Jia, Wei Zhao and Jie Li, "An integrated Routing and Admission Control Mechanism for Real-Time Multicast Connections in ATM Networks," *IEEE Transactions on Communications*, Vol. 49, No. 9, September 2001, pp. 1515-1519.

[John75] Donald B. Johnson, "Finding all the Elementary Circuits of a Directed Graph," *SIAM J. Computing*, Vol. 4, No. 1, March 1975, pp. 77-84.

[Karp72] R. M. Karp, "Reducibility Among Combinatorial Problems," Complexity of Computer Computations, Plenum Press, 1972.

[KoSu02] S. Koo, S. Subramaniam, "Trade-offs between Speed, Capacity and Restorability in Optical Mesh Network Restoration," *Optical Fiber*

Communication Conference and Exhibit (OFC 2002), 17-22 March 2002. Anaheim, California, pp. 487 -489.

[KrLu95] D. Krizanc, F.L. Luccio. "Boolean routing in chordal rings," *Proceed. of 2nd International Colloquium on Structural Information and Communication Complexity*, Carleton University, Ottawa, Canada, 1996, pp. 89-100.

[LEDA06] LEDA v. 5.2, http: //www.algorithmic-solutions.com/, last accessed in Aug 2006

[LiWa01] H.-C. Lin, C.-H. Wang, "A hybrid multicast algorithm for single-hop WDM networks," *J. of Lightwave Technology*, 2001, Vol. 19, No. 11, pp. 1654-1664.

[LoCh03] Chi-Chun Lo, Bin-Wen Chuang, "A Novel Approach of Backup Path Reservation for Survivable High-Speed Networks," *IEEE Communications Magazine*, Vol. 41, Issue: 3, March 2003. pp. 146 -152.

[LuAn02] Y. Luo, N. Ansari, "Restoration with wavelength conversion of WDM networks," *Electronics Letters,* Vol. 38 Issue: 16, 1 Aug. 2002. pp. 900 -901.

[LuMe01] S.S. Lumetta, M. Medard , "Towards a Deeper Understanding of Link Restoration Algorithms for Mesh Networks," INFOCOM 2001. *Proceed. of IEEE Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, 22-26 April 2001. Alaska, USA, Vol. 1, pp. 367 - 375.

[MaDe76] Prabhakar Mateti and Narsingh Deo. "On algorithms for enumerating all circuits of a graph," *SIAM J. on Computing*, Vol. 5, No. 1, March 1976, pp. 90-99.

[Maie02] Guido Maier, Achille Pattavina, Simone De Patre, Mario Martinelli, "Optical network survivability: protection techniques in the WDM layer," *Photonic Network Communications,* Vol. 4, Issue 3, July 2002 - December 2002, pp. 251-269.

[MaBe93] K. Makki, K. Been, N. Pissinou, "A Parallel Algorithm for the Steiner Tree Problem," *Proceed. of IEEE Fifth International Conference on Computing and Information 1993* (ICCI '93), Sudbury, Ont., Canada, pp. 380 – 384.

[MaLe02] M. Ajmone Marsan, E. Leonardi, M. Mellia, A. Nucci and A. Grosso, "Design of Logical Topologies in Wavelength-Routed IP Networks," *J. of Photonic Network Communications*, July-Dec 2002, pp. 423-442

[MAPN05] MAPNET, http://www.caida.org/tools/visualization/mapnet/. Information accessed in 2005.

[MaYa04] Wail Mardini, Oliver Yang and G.Q. Wang, "Mapping P-Cycles From Planar Graph to Cycle Graphs," CISS 2004. *Proceed. of 38th Annual Conference on Information Sciences and Systems*, 17-19 March 2004. Princeton, NJ, USA.

[MeFi99] M. Medard, S. G. Finn, R. A. Barry, R. G. Gallager, "Redundant Trees for Pre-planned Recovery in Arbitrary Vertex-Redundant or Edge-Redundant Graphs," *IEEE/ACM Transactions on Networking*, Vol. 7, Issue 5, Oct.1999, pp. 641 - 652.

[MoGr99] D. Morley and W.D. Grover, "Current Approaches in the Design of Ring-based Optical Networks," *Proceed. of IEEE CCECE 1999*, Edmonton, Canada, May 1999, pp. 220-225

[MoSh91] B. Moret and H. Shapiro. "*Algorithm from P to NP: Design and Efficiency,*" Benjamin/Cummings, Redwood City, CA, 1991.

[OhNo90] Naohisa Ohta, Mitsuru Nomura, Nobuyuki Tokura and Katsuaki Kikuchi, "Video distribution on ATM-based optical ring networks," *Proceed. of IEEE International Conference on Communication*, ICC'90, 16-19 Apr 1990, Atlanta, GA, USA, pp. 976-980.

[Ozda03] Asuman E. Ozdaglar, "Routing and Wavelength Assignment in Optical Networks," *IEEE/ACM Transactions on Networking*, Vol. 11, No. 2, April 2003.

[QiJe99] C. Qiao, M. Jeong, A. Guha, X. Zhang, J. Wei, "WDM multicasting in IP over WDM networks," *Proceed. of 7th International Conference on Networks Protocols*, October 31 - November 03, 1999, pp. 89-96.

[RaMu99a] S. Ramamurthy, B. Mukherjee, "In Survivable WDM Mesh Networks, Part I – Protection," *Proceed. of IEEE INFOCOM*, Vol. 2, pp. 744 -751, 1999.

[RaMu99b] S. Ramamurthy, B. Mukherjee, "In Survivable WDM Mesh Networks, Part II – Restoration," *Proceed. of International Conference on Communication 1999* (ICC 1999), Vancouver, June 1999, pp. 2023-2030.

[Rayw83] V. J. Rayward-Smith, "The Computation of Nearly minimal Steiner Trees in graphs," *International J. of Mathematical Education in Science and Technology*, Vol. 14. No. 1, 1983, pp. 15-23.

[RuLu02] Lu Ruan, Haibo Luo, "A Fast Lightpath Restoration Method Using Two Backup Paths in WDM Networks," *Proceed. of International Conference on Parallel Processing Workshops*, 2002. 18-21 Aug. 2002, pp. 183 -189

[SaAz98] G. Sahin, M. Azizoglu, "Multicast routing and wavelength assignment in wide area networks," *Proceed. of SPIE*, Vol. 3531, pp. 196-208, November 1998.

[SaMu00] L. H. Sahasrabuddhe, and B. Mukherjee, "Multicast Routing Algorithms and Protocols: a Tutorial," *IEEE Network*, January/February 2000, pp. 90-102.

[ScSe05] Michael Scheutzow, Patrick Seeling, Martin Maier Martin Reisslein, "Multicast Capacity of Packet-Switched Ring WDM Networks," *Proceed. of 24th Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 1, 13-17 March 2005, pp.706 – 717.

[ScGr02] D.A. Schupke, C. G. Gruber, A. Autenrieth, "Optimal configuration of p-cycles in WDM networks Communications," *Proceed. of International Conference on Communication 2002* (ICC 2002). Vol. 5, 28 April-2 May 2002, pp. 2761 – 2765.

[Schu03] D.A. Schupke, "The tradeoff between the number of deployed p-cycles and the survivability to dual fiber duct failures," *Proceed. of International Conference on Communication 2003* (ICC 2003). 11-15 May 2003, Vol. 2, pp. 1428- 1432.

[ShLu95] A. Shaikh, S. Lu, K. Shin, "Localized Multicast Routing," *IEEE GLOBECOM'95*, Singapore, Nov. 1995, pp. 1352-1356.

[ShYa01] Shahram Shah-Heydari and Oliver Yang, "A tree-based algorithm for protection/restoration in optical mesh networks," *Electrical and Computer Engineering, 2001 Canadian Conference,* Vol. 2, 13-16 May 2001 pp. 1169 – 1174.

[ShYa04] Shahram Shah-Heydari, Oliver Yang, "Hierarchical Protection Tree Scheme for Failure Recovery in Mesh Networks," *Photonic Network Communications*, Vol.7, No. 2, March 2004, pp. 145-159.

[StGr00] D. Stamatelakis, W. D. Grover, "Theoretical Underpinnings for the Efficiency of Restorable Networks Using Preconfigured Cycles ("p-cycles")," IEEE Transactions on Communications, Vol. 48, No. 8, pp. 1262-1265, August 2000.

[StGr99] D. Stamatelakis, W. D. Grover, "Network restorability design using pre-configured trees, cycles and mixture of pattern types," TRLabs Technical Report TR-1999-05, Issue 1.0, October 30 2000.

[Sysl81] Maciej M. Syslo, "An efficient cycle vector space algorithm for listing all cycles of a planar graph," *SIAM J. on Computing*, Vol. 10, No. 4, pp. 797-808, November 1981.

[Sysl82] Maciej M. Syslo, "On the Fundamental Cycle Set Graph," *IEEE Transactions On Circuits And Systems*, Vol. CAS-29, No. 3, March 1982

[TaMa80] H. Takhasi, M. Matsuyama, "An Approximate Solution for the Steiner Problem in Graphs," *Mathematica Japonica*, Vol. 24, No. 6, 1980, pp. 573-577.

[Tarj73] Robert Tarjan, "Enumeration of the elementary circuits of a directed graph," *SIAM J. on Computing*, Vol. 2, No. 3, pp. 211-216, September 1973.

[Tier70] James C. Tiernan, "An efficient search algorithm to find the elementary circuits of a graph," *J. Communication ACM*, Vol. 13, pp. 722-726, 1970.

[ToNe94] M. To, P. Neusy, "Unavailability analysis of long-haul networks," *IEEE J. on Selected Areas in Communication*, Vol.12, No.1, January 1994, pp. 100-109.

[TsTs01] Cheng-Fa Tsai, Chun-Wei Tsai and Han-Chan Wu, "A Novel Multicast Routing Algorithm with Delay Constraint for Multimedia Applications in Wide Area Network," *Proceed. of International Conferences on Info-tech and Info-net, 2001* (ICII 2001), Vol. 5, Beijing. October 29- November 1, 2001, pp. 301-305.

[TsKu00] W. Y. Tseng, S. Y. Kuo, "Hybrid scheduling for unicast and multicast traffic in broadcast WDM networks," IEICE Transactions on Communications, Vol. E83-B, No. 10, Oct. 2000.

[VePo02] A. J. Vernon, J. D. Portier, "Protection of Optical Channels in All-Optical Networks," Proceed. of 18th Annual National Fiber Optic Engineers Conference (NFOEC 2002), pp. 1695-1706, Dallas, TX, Sept.2002.

[Wase91] Ondria J. Wasem, "An Algorithm for Designing Rings for Survivable Fiber Networks," IEEE Transactions on Reliability, Vol. 40, No. 4, Oct 1991, pp. 428-432

[Weis06] Eric W. Weisstein. "Chord." From MathWorld--A Wolfram Web Resource. http: //mathworld.wolfram.com/Chord.html

[Wils72] Wilson, R. J., "Introduction to Graph Theory," Academic Press, New York (1972).

[Yate99] J. M. Yates, "Wavelength Converters in Dynamically-Reconfigurable WDM Networks, IEEE Communications Surveys Second Quarter (1999) http: //www.comsoc.org/pubs/surveys.

[ZhWe00] X. Zhang, J. Wei, C. Qiao, "Constrained multicast routing in WDM networks with sparse light splitting," *J. of Lightwave Technology*, Vol. 18, No. 12, December 2000. 1917.

[ZhHu93] J. Zhongqi, S. Huaying, G. H. Ming, "Using Neural Networks to Solve Steiner Tree Problem," Proceed. of IEEE TENCON'93, 1993, pp. 807 – 810.

[ZhQi99] X. Zhang, J. Wei, C. Qiao, "On fundamental issues in IP over WDM multicast," Proceed. of 8th IEEE International Conference on Computer Communications and Networks 11–13 Oct (1999) Boston.

[ZhYa02a] Hanxi Zhang, Oliver Yang, "Finding Protection Cycles in DWDM Networks," Proceed. of IEEE International Conference on Communications (ICC 2002), Vol. 5, No. 28, April-2 May 2002, pp. 2756 – 2760.

[ZhYa02b] Yuna Zhang, O. W. W. Yang, "A distributed tree algorithms for WDM network protection/ restoration," 5th IEEE International Conference on High Speed Networks and Multimedia Communications, 3-5 July 2002. pp. 289 -294

[ZhYa02c] Yuna Zhang, Oliver Yang, "Different implementations of token tree algorithm for DWDM network protection/restoration," Proceed. of Computer Communications and Networks, 2002, pp.290-295

[ZhZh03] W. Zhang, D. Zhong, "Design of survivable WDM network using p-cycles," Proceed. of Conference on the Optical Internet (COIN) 2003 and 28th Australian Conference on Optical Fibre Technology (ACOFT 2003), Melbourne, Australia, 13-16 Jul. 2003, pp. 465-468.

Appendix A:
Mapping Between Regular Graph and Modified Cycle Graph

In the following we present a detailed study for the cases of two and three adjacent faces in both regular and dual domain explaining the cases at which we have PCCs. Summary of these cases have been presented in Chapter Three.

A1. Regular Domain

One face is a simple cycle with no straddling links (then it is PCC). Figure 3.1a shows two adjacent faces with one edge in common. This is also PCC with one chord link which is the common edge between the two faces. However, two adjacent faces with more than one edge in common cannot be one PCC (Figure 3.1b). An exception for this happen when one of the two faces has one edge that is not common with the other face at which case we could have a PCC (Figure 3.1c).

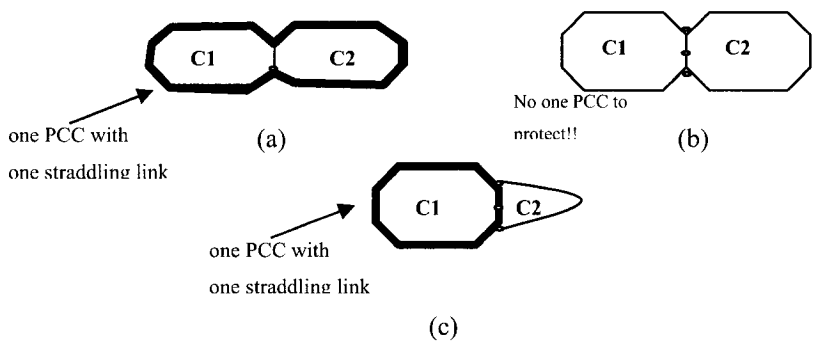


Figure A.1: Different Cases of Two Adjacent Cycles

For the case of three adjacent faces, let us first define some variables to characterize various scenarios:

a_{ij}: the number of edges common between face i and j. $a_{ij} \geq 0$.

b_i: the number of unshared edges of face i with other faces. $b_i \geq 1$.

- Assume that we have three faces labeled 1, 2 and 3. We can use the above variables to classify all the possible cases of three faces for which a one PCC exists that uses all the links of those faces (i.e. either on-cycle or chords):

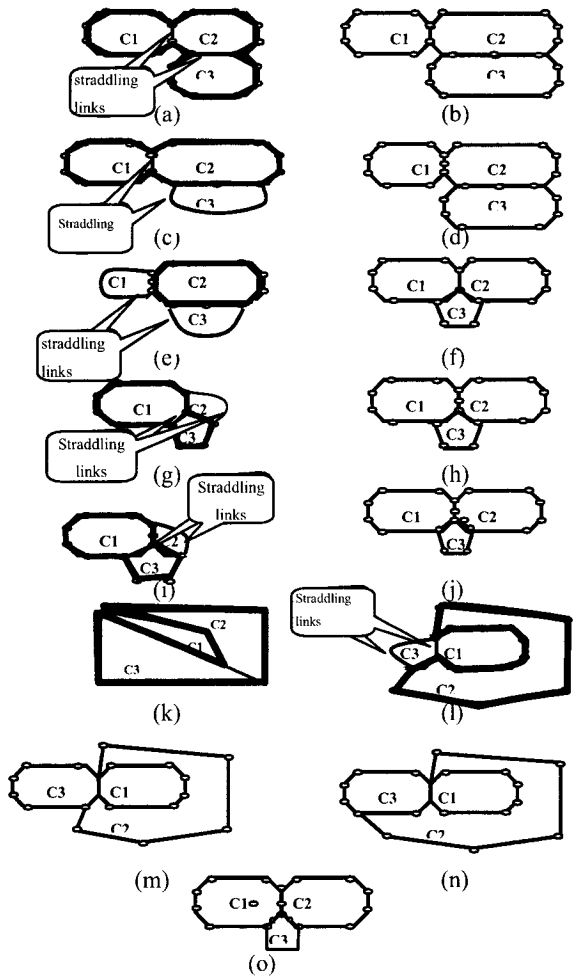


Figure A.2: Different Cases of Three Adjacent Cycles

- $a_{12} = 1$, $a_{23} = 1$, $a_{13}=0$: Figure A.2a shows a PCC exists regardless the value of bi's.
- $a_{12} = 1$, $a_{23} > 1$, $a_{13}=0$: Figure A.2b shows that no PCC exists unless $b_3=1$ (e.g. Figure A.2c).
- $a_{12} > 1$, $a_{23} > 1$, $a_{13}=0$: Figure A.2d shows that no PCC exists unless both b_1 and $b_3 = 1$ (e.g. Figure A.2e).
- $a_{12} = 1$, $a_{23} = 1$, $a_{13} = 1$: Figure A.2f shows that no PCC exists unless at least one of b_1, b_2 and b_3 equal to 1 (e.g. Figure A.2g with $b_2=1$).
- $a_{12} > 1$, $a_{23} = 1$, $a_{13} = 1$: Figure A.2h shows that no PCC exists unless either b_1 or $b_2 =1$ (e.g. Figure A.2I with $b_2=1$). We could also have $b_1=0$, the condition becomes if b_2 or $b_3 = 1$(Figure A.2k and 3.2m).
- $a_{12} > 1$, $a_{23} > 1$, $a_{13} = 1$: Figure A.2j shows that a PCC exists if $b_2 =1$. We could also have $b_1=0$, and in this case, there is no PCC (Figure A.2n).

$a_{12} > 1$, $a_{23} > 1$, $a_{13} > 1$(Figure A.2o): no PCC exist whatever the value of b_1, b_2 and b_3

Note that many other cases are omitted since they are similar to these cases. They could be obtained from the permutation of subscripts, e.g. ($a_{12} = 1$, $a_{23} > 1$, $a_{13} = 1$) is equivalent to ($a_{12} > 1$, $a_{23} = 1$, $a_{13} = 1$) by changing labels and notice that $a_{ij} = a_{ji}$. (Note that these two cases have the same explanation in words: we have three faces and two pairs of them have one edge in common and one pair has more than one edge in common.)

Summarizing and classifying the cases above, we can have PCC in any of the following cases

1) For the case of two pairs of adjacent faces (a12 and a23 > 0, and a13=0):

a) When both pairs have one edge in common (a12 and a23 =1), no other condition imposed.
b) When one pair has more than one edge in common (a12 =1 and a23 >1), we should have b3= 1.
c) When both pairs have more than one edge in common (a12 and a23 >1), we should have both b1 and b3= 1.

2) For the case of three pairs of adjacent faces (a12, a23 and a13>0):

a) When all the three pairs have one edge in common (a12, a23 and a13 =1), we should have at least any of the b1, b2 and b3 =1.
b) When one pair has more than one edge in common (any of a12, a23 and a13>1. Denote the two subscripts of this a as x and y, the third subscript as z),
1) If two faces of x, y and z are node-jointed at a node other than the pair of nodes where they are edge-jointed, we should have b for any of these two faces equals 1.
2) Otherwise (no two such faces exits), we should have bx or y=1.

c) When two pairs have more than one edge in common (two of a12, a23 and a13 are >1, say axy and ayz>1, and thus axz=1), if there is no two faces that are node-jointed at a node other than the pair of nodes where they are edge-jointed, we require by =1.

Thus, for the case of two pairs of adjacent faces in a group of three faces, a PCC will exist either by having a_{ij}=1 for the two pairs (no more than one edge in common), or by having either b_i or b_j =1 when a_{ij}>1. For the case of three pairs of adjacent faces in a group of three faces, existing as a cycle of faces as defined in Chapter 2, we should apply the conditions of two pairs first. We also require at least on of the $b_{i's}$ to be 1. Another case would be when two faces of the three are node-jointed at a node other than the two nodes where they are edge-

jointed (In Figure A.2k, face C2 and C3 are node-jointed in the upper left node and this node is not part of the edges disjoint these two faces). In such case, if we have one pair of faces that have more than one common edge, the condition of having one of the $b_{i's}$ to be 1 to restrict it to specific subscript to 1, not any i. When we have more than one pair, then no PCC exists.

Figure A.3: Cases of Two Adjacent Faces Represented in the Modified Cycle-Graph

A2. Dual Domain

Observations stated in the previous section in the modified cycle graph are illustrated in Figure A.3 and Figure A.4.

We now examine all the cases of n=2 and 3 nodes in MC(G) that have a potential of forming a PCC in G. These cases correspond to 2 and 3 faces in the cycle space which can be found in the previous section in Figure 3.1 and Figure A.2. Their dual graph representation is presented in Figure A.3 and Figure A.4 (with e(n) of each node shown as number beside each node) from which we can make the following observations for the existence of a PCC:

1) For two connected nodes in MC(G) as in Figure A.3, we consider the following two subcases:

   a) When the two nodes are connected by only one link as in Figure A.3a (which corresponds to the case of two adjacent faces with one edge shared between them), then there is a PCC to cover the links of the two corresponding faces.

   b) When the two nodes are connected by more than one link (which correspond to the case of two adjacent faces with more than one edge shared between tem), then we will have a PCC to cover the links of the two corresponding faces if only either e(C1)=deg(C1)+1 or e(C2)=deg(C2)+1 but not both(i.e. one of the faces has only one other link not common with the other face). For example, Figure A.3b has two links (as it corresponds to Figure 3.1b), However, its e(C1)=e(C2)=9 and deg(C1)=deg(C2)=2, meaning that neither e(C1)=deg(C1)+1 nor e(C2)=deg(C2)+1 is

satisfied. Now one would find that Figure 3.1c would form a PCC because as shown in Figure 3.5c $e(C2)=deg(C2)+1$.

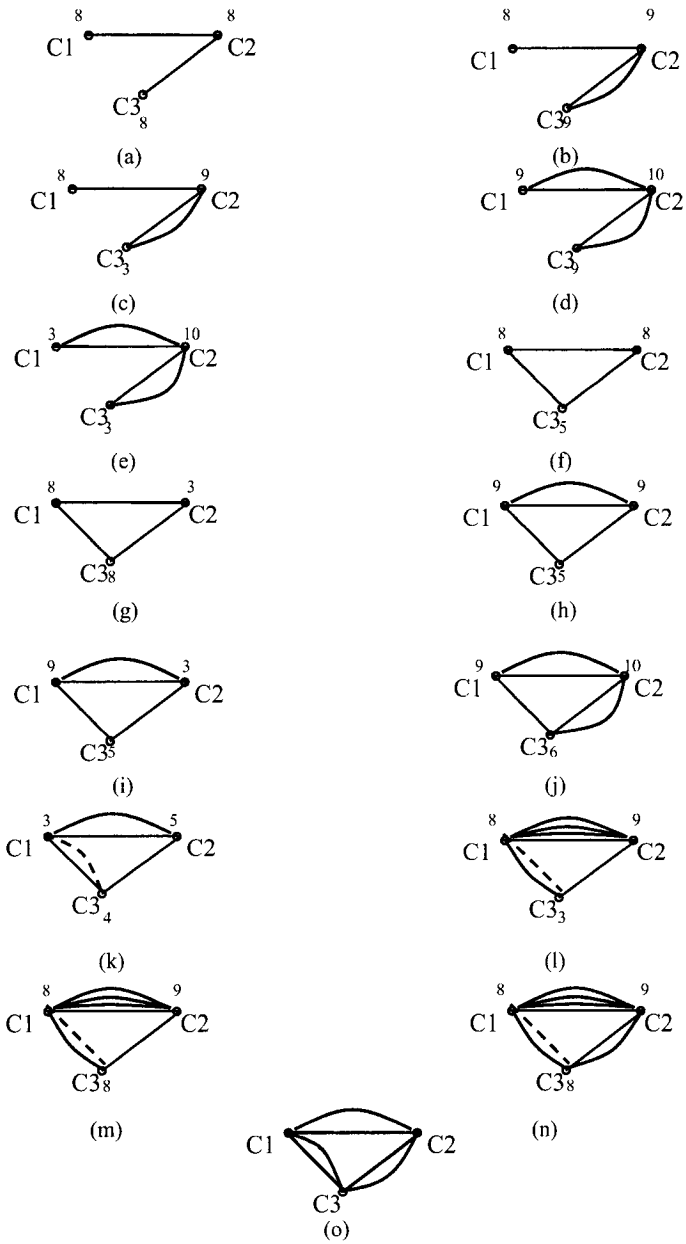


Figure A.4: Cases of Three Adjacent Faces Represented in MCG (e(N) are shown for each Node. The Cases Correspond to Cases in Figure A.2)

2) For the three connected nodes in MC(G) as in Figure A.4, there are also two subcases:

   a) If we do not have a cycle of the three nodes in MC(G), then we must have the following conditions in order for this path in MC(G) to correspond to a PCC in G

      i) If there is exactly one edge between (C1, C2), one edge between (C2, C3), and no

other conditions, then a PCC exist, e.g. Figure A.4a.

ii) If there is more than one edge between (C1, C2), one edge between (C2, C3) (Figure A.4c), and e(C1)-*deg*(C1)=1, then a PCC also exists, e.g. Figure A.4b and Figure A.4c.

b) If we have a cycle for the three nodes in MC(G), then this cycle in MC(G) would correspond to a PCC in G if at least one of the nodes in this cycle (say *v*) should have the property that e(v)-*deg*(*v*)=1. Note that the two nodes other than v should be under the conditions of either 1a or 1b stated above.

For example, in Figs. 6f, 6g, 6h, 6i and 6j, we have a cycle of faces C1, C2 and C3. In Figure A.4f and Figure A.4g we should have any of C1, C2 or C3 with the property e(.)-*deg*(.)=1 and this is the only condition since there is one edge between each pair of the nodes. In Figure A.4h and Figure A.4i, if we have C3 with e(C3)-*deg*(C3)=1, then this will satisfy the condition of three faces. However, the condition of two faces between C1 and C2 is not satisfied because we have two edges between them. If we started by imposing either e(C1)-*deg*(C1)=1 or e(C2)-*deg*(C2)=1, then this will satisfying the condition of two faces between C1 and C2 and in the same time satisfy the condition of three faces (Figure A.4i). In Figure A.4f, we should have C2 with e(C2)-*deg*(C2)=1, then this will satisfy the condition of two faces between C2 and C1 from one side and C2 and C3 from another side, and at the same time satisfy the condition of three faces.

Besides these conditions, a dashed edge is not allowed to form a cycle with the other regular edges as in g, Figure A.4h, Figure A.4i and Figure A.4j. An exception is when we have multiple edges between two nodes (e.g. In Figure A.4g C1 and C2), we could have a dashed edge between one end C1 of the multiple edge and the third node C3, such that either e(C1) - *deg*(C1) = 1 or e(C3) - *deg*(C3) = 1. This face, in fact, is an external face with respect to the PCC formed from the other two faces and still covered by this PCC by having all of its edges except one shared by the on-cycle edge and the other edge as a chord link.

Appendix B:
Example for All-Routes, Rings, and P-cycles

We now present an illustration example by using the network of 7 nodes and 10 edges in Figure B.5. The information is provided here to supplement Section 5.7.

In this example we shall discuss a few cases at which a restriction on the protection path length would affect the optimal value as well as presenting the most optimal solution for the network with different maximum working capacity. We discuss the idea in more details for the three protection methods. In order to explain the idea of pre-configuration and the optimal design, we shall first use a brute-force method to enumerate all paths/cycles. The result for the cycle-based pre-configuration (ring or p-cycle) is shown in Table B-1 while all-paths is show in Table B-2 below. In our comparison discussion below, we assume the capacity of each edge is the cost with the same i units, i.e. $i=F=K$

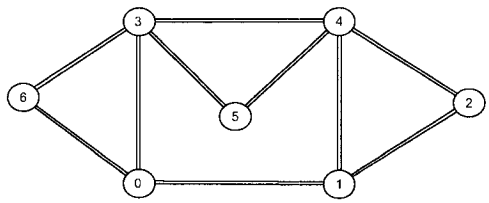


Figure B.5: Graph With 7 Nodes and 10 Edges

Table B-1: Cycle Enumeration of the Network in Figure B.5

1	3-4 3-5 4-5
2	1-2 1-4 2-4
3	0-1 0 3 1-4 3-4
4	0-1 0 3 1-2 2-4 3-4
5	0-3 0-6 3-6
6	0-1 0-3 1-4 3-5 4-5
7	0-1 0-3 1-2 2-4 3-5
8	0-1 0-6 1-4 3-4 3-6
9	0-1 0-6 1-2 2-4 3-4
10	0-1 0-6 1-4 3-4 3-6
11	0-1 0-6 1-2 2-4 3-5

Table B-2: All Paths Enumeration of the Network in Figure B.5

1	1-4 3-4	34	0-1 1-2 2-4 3-4 3-6
2	1-2 2-4 3-4	35	0-1 3-6 3-5 4-5
3	0-6 3-6	36	0-1 1-2 2-4 3-6 3-5 4-5
4	1-4 3-5 4-5	37	1-2 1-4
5	1-2 2-4 3-5 4-5	38	0-1 1-2 3-4
6	0-1 1-4 3-4 3-6	39	0-1 1-2 3-5 4-5
7	0-1 1-2 2-4 3-4 3-6	40	0-1 1-2 3-4 3-6
8	0-1 1-4 3-6 3-5 4-5	41	0-1 1-2 3-6 3-5 4-5
9	0-1 1-2 2-4 3-6 3-5 4-5	42	3-5 4-5
10	0-3 3-6	43	0-1 1-4
11	1-4 3-4 3-6	44	0-1 1-2 2-4
12	1-2 2-4 3-4 3-6	45	0-1 1-4 3-5 4-5
13	1-4 3-6 3-5 4-5	46	0-1 1-2 2-4 3-5 4-5
14	1-2 2-4 3-6 3-5 4-5	47	0-1 1-4 3-6
15	1-4 3-4	48	0-1 1-2 2-4 3-6
16	1-2 2-4 3-4	49	0-1 1-4 3-6 3-5 4-5
17	1-4 3-5 4-5	50	0-1 1-2 2-4 3-6 3-5 4-5
18	1-2 2-4 3-5 4-5	51	0-3 0-6
19	1-4 3-4 3-6	52	0-1 1-4 3-4
20	1-2 2-4 3-4 3-6	53	0-1 1-2 2-4 3-4
21	1-4 3-6 3-5 4-5	54	0-1 1-4 3-5 4-5
22	1-2 2-4 3-6 3-5 4-5	55	0-1 1-2 2-4 3-5 4-5
23	1-4 2-4	56	3-4 4-5
24	0-1 2-4 3-4	57	0-1 1-4 4-5
25	0-1 2-4 3-5 4-5	58	0-1 1-2 2-4 4-5
26	0-1 2-4 3-4 3-6	59	0-1 1-4 3-6 4-5
27	0-1 2-4 3-6 3-5 4-5	60	0-1 1-2 2-4 3-6 4-5
28	1-4 2-4	61	3-4 3-5
29	0-1 3-4	62	0-1 1-4 3-5
30	0-1 1-2 2-4 3-4	63	0-1 1-2 2-4 3-5
31	0-1 3-5 4-5	64	0-1 1-4 3-6 3-5
32	0-1 1-2 2-4 3-5 4-5	65	0-1 1-2 2-4 3-6 3-5 1
33	0-1 3-4 3-6		

B.1 All-Paths

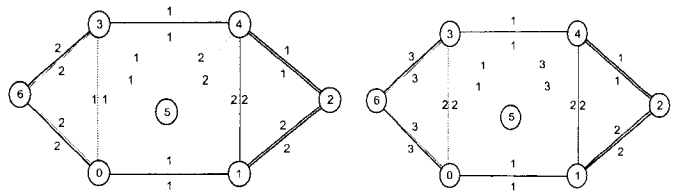
In this approach, knowing all possible paths in the network definitely guarantees the existence of the "most" optimal solution for the network. We first consider the following few cases of enumeration.

For i=1, the optimal set of routes is {#3, #14, #22, #27, #28, #41, #42, #55, #60, #65} with a 7 unit of spare capacity required. Notice that we have 10 routes as preconfigured protection routes for the graph , one for each edge (e.g. link 0-3 can be protected by route #3). Notice also that 7 is the number of different edges in the collection of edges for this set of routes.

For *i*=2, the optimal set of routes that protects the whole network is {#3, #10, #12, #16, #25, #26, #28, #29, #40, #42, #51, #53, #56, #61, #63}. The optimal cost is 20. Notice that the edges of one unit of working capacity needs 1 path with 1 unit of spare capacity on each link (e.g. link 0-3 can be protected by route #3), while the edges of two units of working capacity needs either 1 path with 2 units of spare assigned to its edges, or two paths each with one unit of spare assigned to its edges (e.g. 0-6 is protected by #10 and #12). For the i=3 case, the optimal set of routes that protects the whole network is {#3, #10, #12, #14, #16, #25, #26, #28, #40, #42, #51, #53, #55, #56, #61, #63, #65} with optimal cost is 35 unit cost.

We readily observe that an optimal solution still exists for path length restriction is greater than 5 since the longest path among all the routes above is 5 hops-long. A solution also exists if the path length restriction is 4 or 3, but more costly. Finally, there is no solution if path length restriction is 2 since edge 0-1 cannot be protected by a path less than 3 hops long.

In summary, if we enumerate the paths with a restriction on the path length, we can obtain a smaller set of paths which might result in a less efficient solution (i.e. using more spare capacity).



(a) max is 2 units (b) max is 3 units

Figure B.6: Network in Figure B.5 with Different Units of Working Capacity Assigned to each Edge

B.2 Rings

For a given cycle, each of the links on that cycle has a protection path that uses all the cycle links except for the link being protected. Thus, a set of candidate cycles in the network provides all the links that appears on at least one cycle in that set with a protection path(s). For the ring preconfiguration scheme, we shall try to find a candidate set of cycles that covers all the links on the network by having each link lie on at least one cycle. Using rigs makes the operation easier but with higher spare capacity cost, as observed in the enumeration below.

Using again Figure B.5 and the set of cycles in Table B-1, the optimal solution for i=1 is achieved by using Cycle #3 and Cycle #11. That is, we use Cycle #11 (with 7 links) to protect the 7 links on its on-cycle links, and Cycle #3 (with 4 links) to protect the remaining 3 links. The total spare capacity in this case is 11 units of spare capacity. For i=2, the optimal set are the Cycles #1, #2, #5 and #10. The total cost of spare capacity required is 29 . For i=3, the optimal set is { #1, #2, #5, #7 and #10}, and the total cost of spare capacity required is 49.

B.3 P-cycles

For the p-cycle preconfiguration scheme, we have to find a subset of cycle such that each link on the network either lies on a cycle (like the rings) or straddles on a cycle. Using the same figure and the set of cycles in Table B-1, we can have the following cases. For i=1, the optimal p-cycle design is Cycle #11 with one unit of capacity assigned to each of its on cycle edges. Notice that this cycle has 7 edges and is a Hamiltonian cycle. This can be obtained easily from the list of cycles in Table B-1 without solving the optimization problem. Thus, using this cycle, the optimal cost of the spare capacity required is 7 to protect the 10 units of working capacity on each link.

When i=2 (Figure B.6a), the optimal cycle is again Cycle #11 but with two units of capacity assigned to each of its on cycle edge. This is not so obvious in contrast to the case of i=1, but can be obtained by solving the optimization problem. Thus, the optimal spare capacity required using all cycles are 2 units on each of the 7 links with a total cost equal to 2(2*4+1*3)=22 (4 links with cost 2, and 3 links with cost 1) to protect the 15 units with total cost of 25 (25=2*2*5+5) (2 unit cost more than all-rotes).

For i=3 (Figure B.6 (4 links with cost 2, and 3 links with cost 1) to protect the 15 units with total cost of 25 (25=2*2*5+5) (2 unit cost more than all-rotes).

For i=3 (Figure B.6b), the optimal set of cycles are Cycles #10 (with 1 unit of spare capacity) and Cycle #11 (with 2 units of capacity). So the optimal spare cost =41 which is 6 units of cost more than all-routes!

In all the three cases above, we can see the optimal solution with no path length restriction needs Cycle #11 to be part of the solution which means that if a restriction on the protection path length is imposed, then we cannot achieve the optimal solution in such cases. To explore further, if the restriction of the path length is 4 hops, there is no solution for any *i,* and if the restriction is 5 hopes, then the most optimal solution will be {#2, #5, #6} for *i*=1 with a total cost of 9 units which is 2 units more than the "most" optimal solution with no restriction on the protection path length.

As an intermediate summary, one can easily see that the all-routes scheme is more efficient than the other schemes. However, the number of protection routes is much larger, resulting in more complicated management and switching in case of a failure. We shall now study the general behavior by presenting a simulation study for a set of randomly created networks with different working capacities.