Quasi Borel Cayley Graphs for Ultrafast Information Dissemination in Large and Dense Networks

A Dissertation Presented

by

Jaewook Yu

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

May 2011

Stony Brook University

The Graduate School

Jaewook Yu

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation.

K. Wendy Tang – Dissertation Advisor
Associate Professor, Department of Electrical and Computer Engineering

Eric C. Noel – Dissertation Co-Advisor
Ph.D, AT&T Laboratories

Thomas G. Robertazzi – Chairperson of Defense
Professor, Department of Electrical and Computer Engineering

Yuanyuan Yang
Professor, Department of Electrical and Computer Engineering

Jie Gao
Assistant Professor, Department of Computer Science

This dissertation is accepted by the Graduate School.

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Quasi Borel Cayley Graphs for Ultrafast Information Dissemination in Large and Dense Networks

by

Jaewook Yu

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2011

As modern networks grow quickly in size, and the amount of data to be processed in the network explodes, fast information dissemination and data exchange is gaining strong attention today. In this research, we observe that the underlying topology of a network plays a greater role in determining the efficiency of the networks in terms of information dissemination speed especially for large networks. However, there has been a lack of constructive research encompassing network modeling, performance evaluation, and application of such network models. Therefore, we devote our efforts to identify a network model enabling ultrafast information exchange, recognize issues that may arise when applying such network model in real networks, and provide corresponding solutions for the problems. In regard to these efforts, this dissertation makes the following contributions. We identified Borel Cayley Graphs (BCGs) to be one of the fastest network topologies in information dissemination for large and dense networks. In addition, we showed that BCGs have favorable topological properties including deterministic topology generation, small nodal degree, short average path length, and small diameter. However, besides these superior properties, it has been challenging to use BCG as an underlying topology for real networks because of its lack of size flexibility. To resolve BCG's size limitation, we proposed BCG Pruning and Expansion algorithms that transform the original BCGs into Quasi BCGs in any desired sizes while maintaining the superior properties of the original BCGs. Analytical and simulation results showed that Quasi BCGs exhibit almost the same information dissemination performance and similar topological properties as those of the original BCGs. In addition, we considered wireless sensor networks to demonstrate the potential of BCGs as a real network topology. Specifically, we developed a topology control protocol called BCG Topology Control (BCG-TC) that constructs Quasi BCG network topology in wireless sensor networks. Lastly, we proposed the Dynamic BCG Routing Protocol that allows nodes in a network to update their routing table dynamically as network topology changes over time.

To My Parents, Brother, and Wife

Contents

List of Figures x

List of Tables xiii

Acknowledgments xiv

1 Introduction 1
1.1 Network Models for Ultrafast Information Dissemination . . . 2
1.2 Problems and Approaches . . . 3
1.3 The BCGs for Wireless Sensor Networks . . . 4
1.3.1 Topology Control . . . 4
1.4 Routing . . . 6
1.5 Summary of Contribution . . . 8
1.6 Organization of Thesis . . . 9

2 Preliminary 10
2.1 Graph Terminology . . . 10
2.2 Topological Graph Metrics . . . 12
2.3 Spectral Graph Metrics . . . 13
2.3.1 Graph Laplacian . . . 13
2.3.2 Graph spectrum . . . 15
2.3.3 Algebraic connectivity . . . 17
2.4 Information Dissemination Performance . . . 17
2.4.1 Average consensus protocol . . . 18

3 Network Models 20
3.1 Introduction . . . 20
3.2 Regular Graphs . . . 21
3.2.1 Ring lattices . . . 21
3.2.2 Meshes . . . 22
3.3 Random Graphs . . . 24

3.3.1 Erdös-Rényi's random graphs 24
3.3.2 Watts-Strogatz's small world networks 26
3.4 Cayley Graphs 27
3.5 Borel Cayley Graphs 27
3.5.1 Node representation 28
3.5.2 Connection rule 28
3.5.3 Examples 29
3.6 How to Select BCG Parameters? 30
3.6.1 BCG samples 30
3.6.2 Patterns of bad generators 30
3.6.3 Guideline 32
3.7 Information Dissemination Performance of Network Models . . 33
3.7.1 Setup 33
3.7.2 Performance metrics 34
3.7.3 Results 34

4 Quasi Borel Cayley Graph **41**
4.1 Introduction 41
4.2 BCG Pruning 42
4.2.1 Cut-through rewiring 43
4.3 BCG Pruning: Connectivity 46
4.3.1 Analysis 46
4.3.2 Validation 50
4.3.3 Relation to the graph generators 53
4.4 BCG Pruning: Topological and Spectral Properties 54
4.4.1 Graph generation 54
4.4.2 Diameter 54
4.4.3 Average path length 55
4.4.4 Algebraic connectivity 55
4.5 BCG Pruning: Information Dissemination Performance 57
4.5.1 Setups 57
4.5.2 Evaluation 58
4.6 BCG Random Expansion 61
4.6.1 Terminologies 61
4.6.2 Random expansion 62
4.7 BCG Random Expansion: Topological Properties 62
4.7.1 Setup 65
4.7.2 Diameter 65
4.7.3 Average path length 66
4.8 BCG Random Expansion: Information Dissemination Performance 66

4.8.1 Setup and metrics . . . 66
4.8.2 Performance . . . 66

5 Borel Cayley Graph Topology Control **71**
5.1 Introduction . . . 71
5.2 Borel Cayley Graph Topology Control . . . 72
5.2.1 Assumptions . . . 72
5.2.2 BCG-TC Phase-I . . . 74
5.2.3 BCG-TC Phase-II . . . 75
5.2.4 Neighbor polling order . . . 76
5.3 Performance Evaluation . . . 78
5.3.1 Network connectivity . . . 78
5.3.2 Topological properties . . . 82
5.4 Energy Consumption . . . 83

6 Routing **86**
6.1 Introduction . . . 86
6.2 Vertex Transitive BCG Routing Protocol . . . 87
6.2.1 Routing table . . . 87
6.2.2 Routing table generation . . . 87
6.2.3 Node ID translation . . . 89
6.3 Dynamic BCG Routing Protocol . . . 90
6.3.1 Assumption . . . 90
6.3.2 Dynamic routing table update . . . 92
6.3.3 Dynamic routing table update with CTR . . . 92
6.3.4 Random forwarding . . . 94
6.4 Backward Advertisement . . . 94
6.4.1 Operation . . . 95
6.5 Routing Performance . . . 96
6.5.1 Setups . . . 96
6.5.2 Reachability . . . 96
6.5.3 Average hop count . . . 98
6.5.4 Hop counts distribution . . . 98
6.6 Discussion . . . 100

7 Conclusion **102**
7.1 Summary . . . 102
7.2 Future Work . . . 103
7.2.1 Open issues . . . 103
7.2.2 Applications . . . 104

Bibliography **106**

List of Figures

1.1 Taxonomy of topology control protocols. 5

2.1 Example undirected graph $G(V, E)$. 10
2.2 Degree, distance, and diameter of the sample graph. 11
2.3 Graph spectra and the histograms of different graph families. . 16
2.4 Asymptotic convergence of state values of nodes in random graph $G(n = 20, m = 8)$. 19

3.1 Algorithm for k-regular ring lattice generator. 22
3.2 Realization of k-regular ring lattice with n=*32* and k=2, 4, 8, and 16. 23
3.3 4-Regular toroidal mesh and diagonal mesh. 23
3.4 Realization of a random graph of 40 vertices and mean degree 3. 25
3.5 Algorithm for random graph generator 26
3.6 Connection rule of Borel Cayley Graph. 27
3.7 A 21-node, 4-regular BCG. 29
3.8 Algorithm for average consensus protocol. 34
3.9 Information dissemination performance versus algebraic connectivity (λ_2) for n = 110, 171, 253, and 1081. 37
3.10 Information dissemination performance versus the ratio of spectral radio to algebraic connectivity (λ_n/λ_2) for n = 110, 171, 253, and 1081. 38
3.11 Convergence steps vs. network sizes. 39

4.1 The quasi Borel Cayley graph generation process. 41
4.2 Algorithm for BCG pruning. 43
4.3 Pruning and Cut-Through Rewiring. 44
4.4 Cutting-through multiple edges. 44
4.5 Handling K_3 cycle. 45
4.6 Algorithm for Cut-Through Rewiring. 46
4.7 The nodal isolation probability and the average nodal isolation ratio of resized BCGs. 51

4.8 The graph disconnection probability and the ratio as a function of the normalized amount of pruned nodes γ/n_o 51
4.9 Analytical error $P_E(dis)$ and the Case 2 graph disconnection ratio $\Delta_{Case2}(dis)$ as a function of the normalized amount of pruned nodes γ/n_o. 52
4.10 $P(dis)$ of the resized graphs as a function of γ/n_o for $\alpha = 10, \ldots, 50$. The size of the original BCG $n_o = 5253$. 53
4.11 Comparison of information distribution performance for the original and resized BCGs, toroidal and diagonal meshes, and SWNs as a function of network size. (continue) 59
4.11 Comparison of information distribution performance for the original and resized BCGs, toroidal and diagonal meshes, and SWNs as a function of network size. 60
4.12 Illustration of node injection and edge rewiring in BCG Random Expansion algorithm. 63
4.13 Algorithm for BCG Random Expansion. 64
4.14 Diameter of the expanded BCGs, the original BCGs, toroidal meshes, diagonal meshes, and SWNs. 67
4.15 Average path length of the expanded BCGs, the original BCGs, toroidal meshes, diagonal meshes, and SWNs. 68
4.16 Information dissemination performance of the expanded BCGs, the original BCGs, toroidal meshes, diagonal meshes, and SWNs. 69

5.1 Algorithm for BCG-TC Phase-I. 74
5.2 BCG-TC Phase-II example. 75
5.3 Algorithm for BCG-TC Phase-II. 76
5.4 Algorithm for processing HELLO. 77
5.5 Illustration of $\alpha - 1$ nodes in x's g_1-direction polling order $\vec{x}(g_1)$. 77
5.6 Average node isolation ratio versus the maximum radio range. 79
5.7 Connected network ratio versus the maximum radio range. . . 80
5.8 Average number of isolated nodes in a network. 81
5.9 The ratio of disconnected networks without isolated nodes. . . 81
5.10 Radio range versus average diameter. 82
5.11 Radio range versus the average path length. 83
5.12 Average nodal energy consumption when the consensus has been made. 84

6.1 An example of BCG ($d = 4$) static routing table for node 0. . 88
6.2 Example routing table with the destination row 17 set based on the shortest paths. 89

6.3 Illustration of destination blocking after node f fails using the Dynamic BCG Routing Protocol. 91
6.4 Illustration of topology and routing table of u after performing CTR. 93
6.5 Illustration of BA packet forwarding to the first hop ancestors. 95
6.6 Node u forwards BA packets to its neighbors a, b, and c. . . . 96
6.7 Reachability versus the percentage of node failures. 97
6.8 Average hop count and average shortest path length versus the percentage of node failures. 98
6.9 Hop count distribution for the Dynamic BCG Routing Protocol with Backward Advertisement. 99

List of Tables

3.1 BCG sizes and corresponding parameters. 31
3.2 (t_1, t_2) generating disconnected BCGs (Scenario A). 32
3.3 Generator pairs generating disconnected BCGs (Scenario B). . 32
3.4 Consensus protocol parameters. 33
3.5 Graph families and parameters. 35

4.1 The original sizes n_o of example BCGs and their corresponding BCG parameters p, k, and a. 42
4.2 Notation. 47
4.3 Sizes and parameters of the original BCGs and corresponding target sizes. 55
4.4 Average diameter of the original and resized BCGs. 56
4.5 Average of average path lengths of the original and resized BCGs. 56
4.6 Average algebraic connectivity of the original and resized BCGs. 56
4.7 Sizes and parameters of the benchmark graphs. 57
4.8 The original BCGs and the target sizes. 62
4.9 Parameters of the benchmark BCGs. 64
4.10 Parameters of toroidal meshes, diagonal meshes, and SWNs. . 65

5.1 Simulation parameters. 78
5.2 Parameters for radio model. 84

6.1 Contents of Backward Advertisement packet. 94

Acknowledgments

Looking back over the past years, I am very grateful for all I have received and learned during my PhD studies. There have been many people who have been encouragements and inspirations throughout these years.

I would like to express my sincere gratitude to my advisor, Dr. Wendy Tang, for her guidance, understanding, and patience during my PhD studies. Dr. Tang has always allowed me ample freedom to explore interesting problems and guided me to the right direction. Her insightful comments and constructive criticisms at different stages of my research helped me achieve quality results.

My co-advisor, Dr. Eric Noel, has been always there to give invaluable comments and advices through the years in my PhD study. I am deeply grateful to him for the patient discussions and suggestions on the technical and theoretical details of my works. I am also thankful to him for his thorough and countless revisions of this thesis.

I would like to gratefully thank the committee members, Dr. Thomas Robertazzi, Dr. Yuanyuan Yang, and Dr. Jie Gao for their thoughtful comments on the thesis. I also thank anonymous reviewers for their insightful comments and suggestions on my previous works during the PhD study.

Many friends have helped me stay on the track through these difficult years. Their support and care helped me overcome many worst moments and stay focused on my study. Their suggestions and discussions gave me insights to look problems in different ways. I specially thank Seungyong for his supports and encouragements. I would like to acknowledge my Lab members Dongsoo and Junghun for their discussions and supports in developing many unclear ideas to the concrete and meaningful ones.

Finally, and most importantly, I would like to thank my wife Jiyoung for her support, encouragement, quiet patience and unwavering love. I thank my parents and brother for their faith in me and allowing me to be whom I wanted. It was their patience and trust that gave me the strength to tackle challenges head on. Also, I thank Jiyoung's parents for their encouragement and support over the years.

Chapter 1

Introduction

As modern networks grow quickly in size and require fast data exchange, designing efficient network topology is of great importance today. For example, the topology of Internet affects the Quality of Service (QoS) and throughput [1, 2], that of the wireless sensor networks (WSNs) affects the data fusion rate [3], and that of social networks affects the speed of information propagation among people [4–6]. More recently, as the daily amount of data that the networked clusters of major Internet companies have to process reaches the order of petabytes[1] [7], the underlying logical or physical topology connecting a large collection of processors, servers, and storages is gaining strong attention too [8, 9]. In such large scale, high density, and data intensive network environments, ultrafast information dissemination and data exchange are the key requirements. However, there has been a lack of constructive research encompassing network modeling, performance evaluation, and application of such network models. Therefore, we have explored the following questions:

(i) Are there any specific network models for fast information exchange for large and dense networks?

(ii) What are the issues and solutions associated with applying the identified network model on real networks?

[1] 1 petabytes = 10^{15} bytes.

1.1 Network Models for Ultrafast Information Dissemination

Among many studies on network models, Watts and Strogatz brought insights into key aspects of efficient network topology. In their seminal work [10], they observed that one can transform a network topology from one with extremely slow in information dissemination to a very fast one by adding a small amount of randomness to the network, where the resulting networks are known as *Small World Networks* (SWNs). To see the effects of randomness on network topology and the information dissemination performance, they begin with k-regular ring lattices which are known as one of the slowest information dissemination graphs. Then, they induced randomness on the graphs by randomly rewiring the edges of k-regular ring lattices with some small rewiring probability. They evaluated the information dissemination performance of the resulting graphs using an epidemic model which resembles a distributed networked system. Their experiments revealed that the infectious diseases spread surprisingly faster in the randomized graphs than in the original k-regular ring lattices. In fact, the randomness give the regular (non-random) graphs a small number of long distance edges acting as shortcuts. These shortcuts transform the *large world* where every node is far away to each other (larger diameter and longer average path length) into the *small world* where nodes are tightly connected (small diameter and shorter average path length). Thus, the resultant graphs are named Small World Networks.

More recently in [11], Olfati-Saber utilized the average consensus protocol that solves an average consensus problem in a distributed manner to show the superior information convergence performance of SWNs. Again, in [12], the same author investigated the connection between fast mixing Markov chains and quasi Ramanujan graphs (Ramanujan graphs generated by a randomized algorithm). We observed that the graphs displaying fast information dissemination in the aforementioned articles share a common aspect: the *randomness* in topology.

In particular, we have been inspired by the pseudo randomness of the *Borel Cayley Graphs* (BCGs) [13]. The topology of BCGs looks random with many long distance edges, however, BCGs are algebraically (and hence deterministically) formulated, k-regular, symmetric, and vertex-transitive graphs. The BCGs are known as one of the densest graphs where the graph density is the number of nodes for a given diameter and the number of edges [13]. The properties of BCGs are summarized as small constant degree, short average path length, and small diameter. The small constant degree implies that each node can communicate simultaneously to only a small number of neighbors. The small diameter and short average path length guarantee the information to traverse the entire network within a small number of hops.

To confirm whether BCGs are the right model for fast information dissemination, we began our initial research by performing a comparative study on the topological properties and information dissemination performance of various graph families including regular graphs, random graphs, and pseudo-random graphs. Specifically, we evaluated BCGs, toroidal meshes, diagonal meshes, random graphs, and SWNs over a wide range of network sizes between 100 to 5000 nodes. Results reported in Chapter 3 showed that BCGs yield 2 to 100 times faster information distribution rate than the SWNs with rewiring probability $p_r = 0.01$, 0.1, or 0.2. More importantly, the BCGs showed the best scalability over the range of network sizes considered. Moreover, topological and spectral graph metrics such as a diameter, average path length, and algebraic connectivity confirmed that BCGs can be very efficient in routing too. Based on our results, we conclude that the BCG is a good candidate for ultrafast information dissemination network topology.

1.2 Problems and Approaches

Despite the favorable properties of BCGs, application of BCGs to real networks have been challenging because of their inflexible and discontinuous sizes. In fact, the size of BCG is determined by $p \times k$ where, p is a prime, and k is the smallest positive integer such that $a^k(\mathbf{mod}\ p) = 1$ for $a \in Z_p$ (see Definition 3.2 and Definition 3.3). This BCG's size restriction has posed a strong challenge to the application of BCGs in real networks. Therefore, we devote our efforts to resolving this size restriction of BCGs and demonstrating potential applications of BCGs in real networks.

To solve the size issue of BCGs, we propose BCG resizing algorithms that construct Quasi BCGs of any arbitrary sizes. We use the term Quasi BCGs because, although the resized BCGs are no longer the original BCGs, they share high level of topological similarities. We will discuss these similarities further in Chapter 4. Our graph resizing algorithm consists of two main algorithms: the *BCG Pruning* algorithm and the *BCG Random Expansion* algorithm.

The BCG Pruning algorithm removes nodes from the original BCGs to generate Quasi BCGs of smaller sizes and rewires the broken connections of remaining nodes to maintain connectivity. Since the BCG Pruning algorithm removes nodes from BCGs, our main concern is whether the constructed Quasi BCGs remain connected while maintaining the properties of the original BCGs. Thus, we provide a theoretical groundwork for the connectivity of Quasi BCGs. In particular, we derived an analytical formulae to characterize the disconnection probability of the Quasi BCGs and further validated the formulae through extensive simulations. Both the analytical and simulation results showed that, using the proposed BCG resizing algorithm, the resized BCGs are *almost surely connected*[2] even after $80 \sim 90\%$ of nodes are pruned from the original BCGs.

Moreover, topological analysis revealed that the original and the pruned BCGs share similar structural properties. In fact, the proposed BCG Pruning algorithm preserved the advantageous properties of the original BCGs when 75% or fewer nodes were pruned from the original BCGs. Such superior properties include a small diameter, a short average path length, and a large algebraic connectivity. Furthermore, we also evaluated the information dissemination performance of both the original and resized BCGs with up to 5000 nodes. The results were promising: the information dissemination performance of pruned BCGs is as good as that of the original BCGs.

Unlike the BCG Pruning algorithm, the proposed BCG Random Expansion algorithm expands the original BCGs to Quasi BCGs with larger sizes by adding nodes into the original BCGs and establishing edges between the added nodes and pre-existing nodes. Simulation results showed that the Random Expansion algorithm expands the original BCGs without noticeable penalty to the information dissemination performance, diameter, and average path length.

1.3 The BCGs for Wireless Sensor Networks

Recently, there has been a growing interest for fast information distribution in densely populated wireless sensor networks [14–16]. Fast information distribution in a large and dense network is important in environmental sensing [17], traffic monitoring [18], security, and surveillance applications [19]. To demonstrate the BCGs applied in real networks, we developed topology control protocol that constructs a wireless sensor network topology similar to BCGs.

1.3.1 Topology Control

Due to the ad hoc nature of WSN, its topology constructed without any control is prone to be arbitrary rather than well constructed to provide better performance for WSN applications. Thus, topology control protocols that construct a network topology compensating the limitations of WSNs have gained attention. Although WSNs demand different sets and levels of performance

[2] We say the resized BCGs are almost surely connected if the graphs are connected with a probability larger than or equal to 0.999.

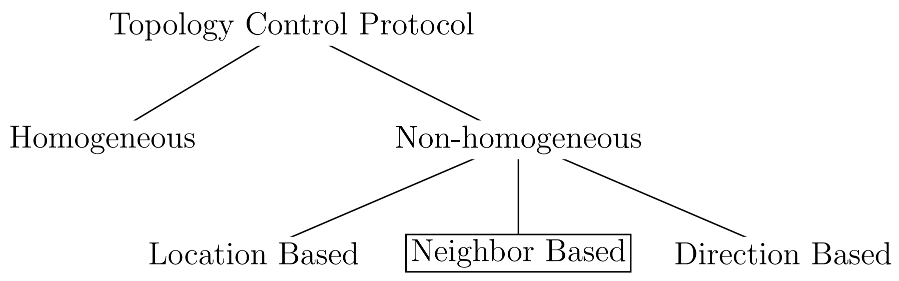


Figure 1.1: Taxonomy of topology control protocols.

requirements depending on applications, the main goals of topology control protocols can be summarized as improving network lifetime, connectivity, information/data fusion speed, and routing efficiency.

Santi [20] provided a comprehensive taxonomy of topology control protocols for wireless ad hoc networks. We show a part of the taxonomy in Figure 1.1. The homogeneous topology control protocols assume that every node has the same radio range, on the other hand, nodes in the non-homogeneous topology control protocols are assumed to control their own radio range within pre-defined maximum radio range. The location based topology control protocols assumes that exact coordinations of nodes are known to a centralized controller, or nodes exchange their locations with neighbors. Based on the location information, the centralized controller may control connections between nodes, or nodes can establish connections with others in a distributed manner. The direction based topology control protocol are similar to the location based topology control protocols in that it utilizes location information. However, nodes in direction based control estimate their relative locations based on their neighboring nodes.

Unlike the location based or direction based topology control protocols, the neighbor based topology control protocols do not rely on the location information of nodes. Instead, the neighbor based topology control protocol assumes that every node uses unique node ID to identify itself and its neighbors within a radio range. Thus, nodes establish connections with their neighbors based on estimated distance or link quality between itself and its neighbors in neighbor based topology control.

In [21], Blough et al. proposed the k-NEIGH topology control protocol that maintains the nodal degree close to but no larger than k as follows: Let L_i be a list of node i's k nearest neighbors. Then, node i collects L_j, where j are the nodes within i's maximum radio range, and builds a set of symmetric neighbors $L^S = \{j \,|\, i \in L_j, j \in L_i\}$. Next, node i sets its transmission power to reach the farthest node in L^S, where L^S is a set of the logical neighbors of i. They showed that nodes of k-Neigh network consume less energy than those of arbitrary network without topology control. They also showed that k-Neigh can generate a connected network with very high probability by setting $k = \Theta(\log n)$, where n is the number of nodes randomly and uniformly distributed in the network. In addition, we developed k-Random, a simple topology control protocol, that randomly selects at most k neighbors from its physical neighbors within maximum radio range R and establishes connections with those selected neighbors.

BCG Topology Control. In our investigation on the topological properties and information dissemination performance of various network models, we conclude that BCGs and Quasi BCGs are one of the best network models for large and dense networks.

However, using the BCGs or Quasi BCGs as a topology for wireless sensor networks has been challenging because the BCG's pseudo-random, long-range connections do not guarantee neighboring nodes to be within radio range. In fact, it is not possible to assume all nodes in a network to be within a single hop communication range in many real network applications. In this sense, we developed the *BCG Topology Control* (BCG-TC) for wireless sensor networks that constructs the network topology as similar as possible to BCG topology while relaxing the one hop communication assumption.

The proposed BCG-TC is a neighbor based topology control protocol that does not rely on the location information of nodes. However, unlike the conventional neighbor based topology control, BCG-TC does not estimate the distance or link quality to establish connections with their neighbor. Instead, nodes in BCG-TC use the pre-loaded, identical BCG parameters to determine their neighbors defined by BCG. Despite the fact that BCG-TC uses pre-defined BCG topology, it still falls into the category of neighbor based topology control. Thus, we compare the performance of BCG-TC with two neighbor based topology control protocols, the k-Neigh and k-Random.

1.4 Routing

Routing protocols for wireless sensor networks (WSNs) have gained growing attention too. In general, sensor nodes of WSNs form a network without the help of central unit (self-organizing network). More importantly, nodes in WSNs frequently die out as they consume their limited energy sources (e.g., batteries). Due to its ad-hoc nature and limited resources, WSN requires a routing protocol that is aware of energy consumption, dynamically changing topolog, network connectivity, node-to-node reachability, and efficiency of data aggregation.

Although BCG networks can use any pre-existing routing protocols such as Ad-hoc On-Demand Distance Vector (AODV) routing protocol [22], the network still requires a routing protocol that exploits favorable properties of the constructed topology. Previously, Tang and Arden in [23] proposed an iterative multi-path routing protocol for BCGs that fully exploits the vertex transitive property of BCGs. Since this routing protocol uses the vertex transitive property of BCG, we call the protocol *Vertex Transitive BCG Routing Protocol*. The Vertex Transitive BCG Routing protocol uses a $(n-1) \times d$ array where n is the number of nodes, and d is a constant degree of BCGs. The routing table contains shortest path information between a root node to the rest of nodes. Since BCGs are vertex transitive graphs, this pre-computed routing table can be used by all other nodes using a simple node ID translation equation. We will discuss the details of the Vertex Transitive BCG Routing Protocol in Chapter 6.

Dynamic BCG Routing Protocol We observed that the static routing table of Vertex Transitive BCG Routing protocol is not suitable for WSNs whose topologies are dynamically changing with time. Thus, we propose the *Dynamic BCG Routing Protocol* that allows nodes in a BCG network to update their BCG routing table as network topology changes.

Initially, routing table of the Dynamic BCG Routing Protocol is the same as that of the Vertex Transitive BCG Routing protocol. It is also a $(n-1) \times d$ array containing multiple shortest path information. However, instead of using the node ID translation, routing tables are pre-translated according to node IDs. Once the translated routing tables are stored at nodes, the routing tables will be independently updated by each node whenever it detects topology (link state) changes. For example, if a node detects one of its neighbors fails, then it updates its routing table to stop forwarding packet destined to the failed node. In addition, we also uses *Backward Advertisement* algorithm for a node to propagate topological changes to its neighbors. Details of the proposed Dynamic BCG Routing Protocol and Backward Advertisement are presented in Chapter 6.

The performance of the Dynamic BCG Routing Protocol has been evaluated in terms of node-to-node reachability, average hop counts, and the distribution of hop counts. When using the Dynamic BCG Routing Protocol without BA, the reachability and average hop counts are degraded as the number of dead nodes increases. However, BA improves the reachability and average hop counts of Dynamic BCG Routing Protocol up to 7% and 35%, respectively, when BA is set to propagate BA packets up to two hop neighbors of a failed node. Propagating BA packets to more neighbors may improve the routing performance, however, our discussion shows that such practice can make BA packets dominate the network bandwidth.

1.5 Summary of Contribution

This dissertation makes the following contributions.

(i) Identified Borel Cayley Graphs (BCGs) to be one of the fastest network topologies in information dissemination for large and dense networks:
We have identified BCGs as one of the most favorable graphs that has many superior properties among various different graph families such as regular ring lattices, random graphs, toroidal meshes, diagonal meshes, and small world networks. The BCGs' favorable properties include deterministic topology generation, small nodal degree, short average path length, small diameter, and, most importantly, ultrafast information dissemination.

(ii) Proposed BCG Pruning and Expansion algorithms that transform the original BCGs into Quasi BCGs in any desired sizes:
Although BCG has superior properties as a network topology, it has been challenging to apply BCG's to real networks because of its lack of size flexibility. We proposed the BCG Pruning and Expansion algorithms to obtain Quasi BCGs with any desired size while preserving the aforementioned superior properties of the original BCGs. Analytical analysis and extensive simulations have confirmed that Quasi BCGs exhibit almost the same information dissemination performance as that of the original BCGs.

(iii) Developed BCG Topology Control (BCG-TC) protocol that constructs Quasi BCG topology for ad hoc networks in a distributed manner:
To show the potential of BCGs as a real network topology, we chose wireless sensor networks. Specifically, we developed a topology control protocol called the BCG-TC that constructs Quasi BCG network topology in wireless sensor networks. We evaluated the performance of BCG-TC in terms of network connectivity, diameter, average path length, and energy consumption. We also showed that BCG-TC generates a connected network with the nodal degree constrained by less than or equal to four, and achieves the least energy consumption among considered topology control protocols such as k-Neigh and k-Random topology controls.

(iv) Developed *Dynamic BCG Routing Protocol* that allows nodes in a BCG network to update their routing tables regarding topology changes:
We proposed Dynamic BCG Routing Protocol based on the BCG's original routing protocol. Each node in a network dynamically updates its routing table using the proposed routing protocol as nodes are dying out. The routing protocol utilizes control message called Backward Advertisement packet to further reduce the number of routing loops and average hop counts, and maximize the reachability.

1.6 Organization of Thesis

Chapter 2 reviews graph terminology, topological and spectral graph metrics, information dissemination performance evaluation techniques used in our research.

Chapter 3 summarizes various network models considered in our research including ring lattices, toroidal meshes, and diagonal meshes, Erdös-Rényi random graphs, and Watts-Strogatz's Small World Networks. This chapter describes details of BCGs including the generation method, connection rules, and parameter selection guidelines. The last part of this chapter is devoted to comparative studies on information dissemination performance of the networks models.

Chapter 4 presents the BCG Pruning and Random Expansion algorithms that transform the original BCGs into the Quasi BCGs with any desired sizes. Analytical and simulation results have been provided to show the topological properties and information dissemination performance of Quasi-BCGs.

Chapter 5 focuses on the application of BCGs on real network applications. In particular, we present the BCG Topology Control (BCG-TC) that constructs a network topology for Wireless Sensor Networks (WSNs). The topological properties of BCG-TC networks including diameter, average path length, and network connectivity have been compared to those of similar topology control protocols.

Chapter 6 proposes the Dynamic BCG Routing Protocol that allows every node in a BCG network to update their routing table as the network topology changes with time. We also present the Backward Advertisement algorithm that propagates topology changes to the neighbors of a failed node. Simulation results showed that the BA improves the proposed routing protocol in terms of source-destination reachability and the average hop counts.

Chapter 7 summarizes our contribution and results. Also, we provide our insights on the future research topics and directions.

Chapter 2

Preliminary

This chapter briefly reviews basic graph terminology, topological graph metrics, spectral graph metrics, and information dissemination performance evaluation techniques used in our research.

2.1 Graph Terminology

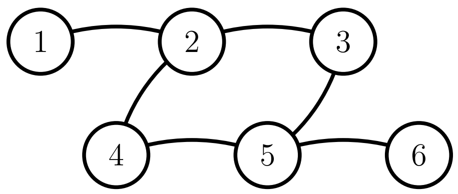


Figure 2.1: Example undirected graph $G(V, E)$.

Definition 2.1 (Graph)**.** A *graph* is an ordered pair $G(V, E)$ such that V is a set of vertices (or nodes) and E is a set of edges connecting the vertices.

Note that vertex and node are used interchangeably throughout this thesis. Similarly, edge and link are used equivalently. The connections of a graph can be represented either by directed edges (directed graph) or undirected edges (undirected graph). Also, it is possible to assign different weights on the edges to make a weighted graph depending on applications or contexts. However, we only consider undirected and unweighted graphs throughout the thesis. For example, an undirected and unweighted graph $G(V, E)$ in Figure 2.1 can be represented by a vertex set V and an edge set E of unordered pair of vertices as follows:

$$V = \{1, 2, 3, 4, 5, 6\} \ ,$$
$$E = \{e_{1,2}, e_{2,3}, e_{2,4}, e_{3,5}, e_{4,5}, e_{5,6}\} \ ,$$

where $e_{u,v}$ is an edge between nodes $u, v \in V$. The *order* of a graph is the number of nodes in a graph.

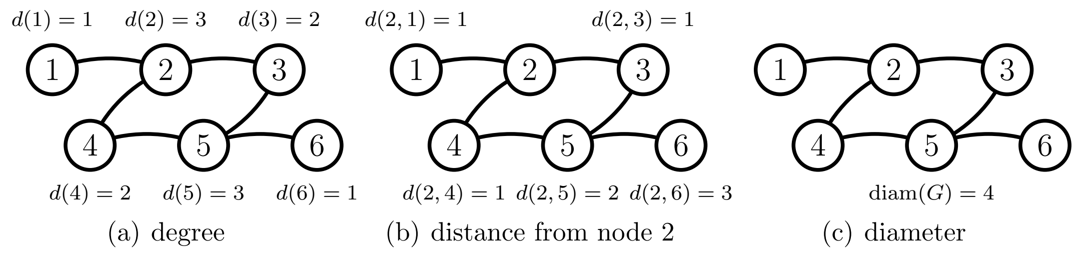


Figure 2.2: Degree, distance, and diameter of the sample graph.

Definition 2.2 (Degree)**.** The *degree* of a vertex $d(v)$ is the number of edges incident to the vertex $v \in V$ (see Figure 2.2(a)).

If $d(v) = k$, $\forall v \in V(G)$, $k > 0$, then G is a *k-regular graph.* A node v is *isolated* if and only if $d(v) = 0$. The *minimum degree* of a graph $d_{\min}(G) := \min\{d(v)|v \in V\}$ and similarly, the *maximum degree* $d_{\max}(G) := \max\{d(v)|v \in V\}$.

Definition 2.3 (Path)**.** A *path* is a graph with a sequence of distinct vertices such that each vertex is connected to the next vertex by an undirected edge.

If a graph H with a vertex sequence $V = \{u, ..., v\}$ is a path, then we say there exists a path between u and v.

Definition 2.4 (Path length)**.** A path length is the number of edges in the path.

A *shortest path* is a path between a pair of nodes of which path length is the smallest. A *shortest path length* is the length of a shortest path.

Definition 2.5 (Distance)**.** A *distance* between two vertices $u, v \in V$, $d(u, v)$, is the length of the shortest path between u and v (see Figure 2.2(b)).

Definition 2.6 (Diameter)**.** A *diameter* of G, diam(G), is the greatest distance between any two vertices in G and denoted by (see Figure 2.2(c)).

Definition 2.7 (Connected graph)**.** A graph is a *connected graph* if there exists a path between any two distinct nodes in a graph.

A *disconnected graph* is a non-empty graph that is not a connected graph.

Definition 2.8 (Component)**.** A maximal connected subgraph of G is a component of G.

A *giant component* is a connected subgraph of G that consists of the majority of nodes in a graph. A non-empty graph G is connected if it consists of only one component. Similarly, G is a disconnected graph if G consists of more than one component.

Definition 2.9 (Vertex transitive graph)**.** A *vertex transitive graph* is a graph such that every vertex has the same local connections, so that no vertex can be distinguished from any other based on the vertices and edges that connect the vertices.

2.2 Topological Graph Metrics

We use various topological graph metrics to characterize properties of graphs under our investigation. In addition to a diameter, degree, and path length, we also use the following metrics.

Definition 2.10 (Average path length)**.** The *average path length* of graph $G(V, E)$ is the expected length of the shortest path between any two vertices.

Let $d(u, v)$ be a distance between two vertices $u, v \in V$ connected by an edge $e_{u,v} \in E$, then the average path length of graph $G(V, E)$ is the average of distances between all possible $n(n-1)/2$ pairs of vertices:

$$\mu(G) = \frac{2}{n(n-1)} \sum_{u,v \in V} d(u, v) \ , \tag{2.1}$$

where $n = |V|$ and $d(u, v) = 0$ if $e_{u,v} \notin E$, or $u = v$.

Definition 2.11 (Degree distribution)**.** The *degree distribution* of a graph is the discrete probability distribution of degree $p = p(k)$.

The degree distribution of a graph represents the probability that a randomly chosen vertex has degree k. Network models represented in graphs can be characterized by their degree distribution, for example, a k-regular graph of which all vertices have the same degree k has a degree distribution following the Dirac delta function given by $\delta(k)$. On the other hand, the degree distribution of a random graph has a binomial distribution which converges to a Poisson distribution as the size of the graph grows [24]. More recently, many articles have reported that the degree distribution of real world networks can be characterized by a power-law distribution [25–30].

2.3 Spectral Graph Metrics

Topological graph metrics such as *degree*, *distance*, *eccentricity*, *centrality*, *diameter*, *clustering coefficient*, and *degree distribution* have been used to capture the structural properties of networks [10, 25, 31, 32]. However, such topological graph metrics often fail to convey the structural properties of a graph as a whole [33]. On the contrary, spectral graph theory has emerged as a comprehensive tool offering a bird's eye view of a graph structure [12, 33–36]. In other words, the whole graph structure can be characterized by a graph spectrum, that is, the set of eigenvalues of a graph Laplacian [37]. In the rest of this section, we review the definitions and interpretations of graph Laplacian, graph spectrum, and algebraic connectivity.

2.3.1 Graph Laplacian

A *graph Laplacian* is the Laplacian matrix of a graph. The tight relationship between the graph Laplacian and the structural properties of a graph has been discussed in [35, 37]. In spectral graph analysis, both eigenvalues of the graph Laplacian and adjacency matrix are referred to as graph spectrum. In our spectral graph analysis, we only consider the spectrum of graph Laplacian as a graph spectrum.

Definition 2.12 (Adjacency matrix)**.** Let $A(G)$ be the *adjacency matrix* of G with n vertices. Then, $A(G)$ is a $n \times n$ binary matrix with its off-diagonal entry $a_{u,v} = 1$ if there exists an edge between u and v, and zeros, otherwise.

We can rewrite each element of $A(G(V, E))$ as follows:

$$a_{u,v}(G) = \begin{cases} 1 & \text{if } e_{u,v} \in E(G), \\ 0 & \text{otherwise,} \end{cases}$$

where $u, v \in V(G)$. The adjacency matrix of an undirected graph with n vertices is a $n \times n$ symmetric matrix since $e_{u,v} \in E(G) \Leftrightarrow e_{v,u} \in E(G)$. For example, the adjacency matrix of the graph in Figure 2.1 is given by

$$A(G) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Definition 2.13 (Degree matrix)**.** Let $D(G)$ be the *degree matrix* of G with n vertices. Then, $D(G)$ is a $n \times n$ diagonal matrix with the diagonal entry $a_{u,u} = d(u), u \in V(G)$, and off diagonal entries are zero.

Definition 2.14 (Laplacian matrix[1])**.** Let $L(G)$ be the *Laplacian matrix* of a graph G. Then, L is $n \times n$ matrix whose off-diagonal entries $l_{u,v} = -1$ if there exists an edge between u and v and 0 otherwise, and whose diagonal entries $l_{u,u} = d(u)$.

By definition of the Laplacian matrix, each element of a Laplacian matrix is given by

$$l_{u,v}(G) = \begin{cases} d(u) & \text{if } u = v, \\ -1 & \text{if } e_{u,v} \in E(G), \ \ u \neq v, \\ 0 & \text{if } e_{u,v} \notin E(G), \end{cases}$$

for all $u, v \in V(G)$. We can rewrite the Laplacian matrix of G using $D(G)$ and $A(G)$ as follows:

$$L(G) = D(G) - A(G). \tag{2.2}$$

[1] Note that, a variant of this definition called the normalized Laplacian matrix appears in some contexts [36].

For example, we obtain the Laplacian matrix of the graph in Figure 2.1 as

$$L(G) = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{D(G)} - \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}}_{A(G)}$$

$$= \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 \\ 0 & -1 & 2 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & -1 & 0 \\ 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}.$$

Furthermore, each row of the Laplacian matrix corresponds to the discrete Laplacian differential operator defined as follows:

$$\Delta\phi_u = \sum_{v\,:\,d(u,v)=1} \Big\{ \phi(v) - \phi(u) \Big\} \ , \tag{2.3}$$

where $d(u, v)$ is the distance between vertices u and v on a graph is the graph Laplacian matrix [38]. In graph theory, the Laplacian matrix defines the relationship between a node $u \in V$ and its immediate neighbors $\mathcal{N}_u = \{v : d(u, v) = 1\}$. We will see the average consensus protocol described in Section 3.8 is tightly related to the Laplacian matrix and corresponding Laplacian operator.

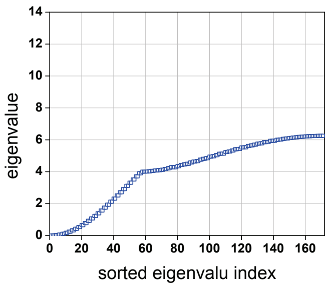
2.3.2 Graph spectrum

The formal definition and some useful properties of graph spectrum are as follows.

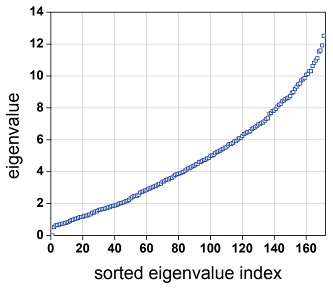
Definition 2.15 (Graph spectrum)**.** A graph spectrum is the set of eigenvalues associated with the Laplacian matrix of a graph.

Let λ_k be the k_{th} eigenvalue of L, and n be the order of a graph, then the ordered eigenvalues of L are denoted by

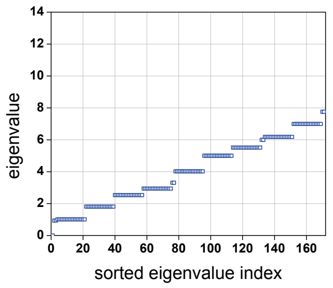
$$0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n.$$



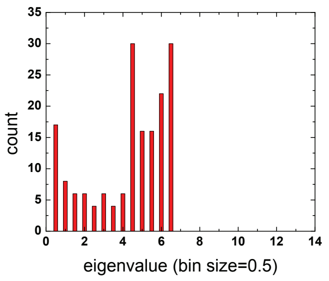
(a) 4-regular ring lattice



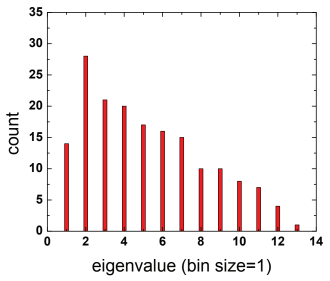
(b) random graph



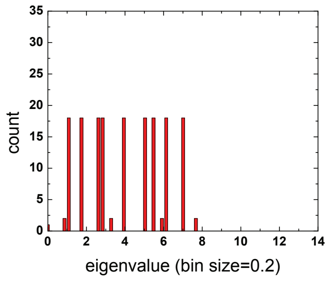
(c) BCG (degree=4)



(d) 4-regular ring lattice



(e) random graph



(f) BCG (degree=4)

Figure 2.3: Graph spectra and the histograms of different graph families.

Since L is positive semidefinite, all of its eigenvalues are nonnegative. The smallest eigenvalue λ_1 is always 0 because $L\mathbf{1} = \mathbf{0}$, where $\mathbf{1}$ denotes the all ones column vector. The graph spectrum has been proven to be a useful tool for graph structure analysis [34, 35], graph clustering [39, 40], graph visualization [41], graph modularizing detection [31], and consensus protocol [42]. In our research, we have obtained a graph spectrum from the graph Laplacian with which we analyzed the structural properties and the information dissemination performance of a graph. In [33], the author reported the existence of unique graph spectra for well-known graph families such as lattices, Erdös-Rènyi's random graphs [43], Watts-Strogatz's small world networks [10], and Barábasi-Albert's scale-free networks [25]. The study showed that graphs belonging to the same family usually share the same or similar spectral properties.

Figure 2.3 shows the shapes of graph spectra for different graph families: k-regular ring lattices, Erdös-Rényi random graphs, and Borel Cayley graphs. The k-regular ring lattice has a spectrum with a saddle point (Figure 2.3(a)) that corresponds to a few nearly identical eigenvalues, and a "bathtub" shaped spectral histogram (Figure 2.3(d)) with a significant concentration of low and high valued eigenvalues. While, the spectrum of the random graph

(Figure 2.3(b)) increases monotonically without any saddle point. Finally, the BCG spectrum (Figure 2.3(c)) resembles that of a “stair case” with nearly identical eigenvalues in each step and a few outliers, which corresponds to a “rectangular” shaped spectral histogram (Figure 2.3(f)).

2.3.3 Algebraic connectivity

Definition 2.16 (Algebraic Connectivity [34])**.** The *algebraic connectivity* of a graph is the second smallest eigenvalue of a graph Laplacian.

The number of zeros that appear in the eigenvalues of a graph Laplacian corresponds to the number of connected components in a graph. If the algebraic connectivity of a graph is zero, then the graph consists of at least two connected components. Thus, G is connected if and only if the second smallest eigenvalue $\lambda_2 > 0$.

Moreover, the magnitude of λ_2 is a measure of how well connected a graph is [34, 37]. Also, it is well known that the larger the algebraic connectivity of a graph, the faster the information dissemination over the graph [12, 44].

Definition 2.17 (Spectral radius)**.** The *spectral radius* of a graph is the largest eigenvalue of the Laplacian matrix of the graph.

In addition, it has also been reported that the ratio of spectral radius to algebraic connectivity, λ_n/λ_2, is another effective measure of information dissemination performance [45]. In our research, we compare the algebraic connectivity of the original and resized BCGs to show how our graph resizing algorithm affects the connectivity and information dissemination performance of the original BCGs.

2.4 Information Dissemination Performance

We consider the *consensus protocol* as a means of evaluating the information dissemination performance of graphs. Consensus protocol is a distributed node-to-node message exchange rule to drive nodes in a network to a network-wide agreement over a quantity of interest (e.g., average of sensory data) [42]. The consensus protocol is widely accepted as a reliable measure of information dissemination or data fusion performance of network topologies [46, 47]. In addition, the consensus protocol has been appeared in the contexts of sensor fusion [46, 48], vehicle formation control [49], and spacecraft attitude control [50–52].

The consensus value can be the average, the maximum, the minimum, or any other function. In particular, we consider the *average consensus protocol* that solves the average consensus problem in a distributed manner. The theoretical background of the average consensus protocol, including asymptotic convergence analysis and its relationship to a graph Laplacian and information dissemination rate, has been studied in [42].

2.4.1 Average consensus protocol

Let us consider a networked system of which the underlying network topology is represented by an unweighted, undirected graph G. Every node in the network holds its own state value (e.g., local temperature data in wireless sensor network application) and updates the state value based on the average consensus protocol. That is, each node in the network, $v \in V(G)$, communicates its state value x_v with its immediate neighbors $N(v) := \{u : e_{u,v} \in E(G)\}$, where $e_{u,v}$ is an undirected edge between u and v, and $E(G)$ is the edge set of G. At each τ_{th} iteration, nodes exchange their state values $x_v(\tau)$ with their immediate neighbors. Given the state values $x_u(\tau)$ received from their neighbors $u \in N(v)$, each node v updates its state according to,

$$x_v(\tau+1) = x_v(\tau) + \frac{1}{\omega} \sum_{u \in N(v)} \Big(x_u(\tau) - x_v(\tau) \Big) \ , \tag{2.4}$$

where $0 < \frac{1}{\omega} < \frac{1}{2d_{max}(G)}$ ensures asymptotic convergence of above average consensus protocol. The asymptotic convergence analysis is well established in [42], and will not be repeated here.

Using the simple average consensus protocol in Eq. (2.4), the state value of each node asymptotically converges to the average of all initial state values. The global agreement or consensus will be asserted when all nodes in the network reach the same average value. Determining the global agreement requires a centralized controller that monitors the state values updated by the nodes in a network. Although this is unrealistic especially for distributed computing applications, we use this protocol solely for measuring the information dissemination performance of networks.

Following the method in [11, 12], we utilize the average consensus protocol, Eq. (2.4), to measure the information dissemination performance of a graph generated by our proposed graph resizing algorithms. This method has been widely accepted as a reliable measure of network information dissemination performance [46, 47]. More details on average consensus protocol can be found in [42, 53] and will not be repeated here.

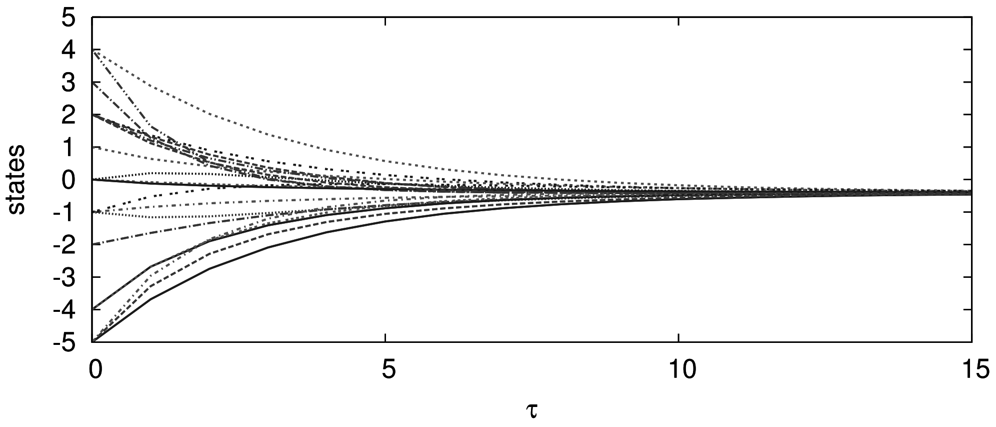


Figure 2.4: Asymptotic convergence of state values of nodes in random graph $G(n = 20, m = 8)$. The consensus value was -0.4.

Example. We start by generating a connected random graph of 20 nodes with a mean degree of 8 where the initial integer state value of each node was chosen uniformly randomly between $[-5, 5]$. The realized random graph had topological metrics, $\mathrm{diam}(G) = 12$, and $d_{max}(G) = 16$. The average consensus value calculated from the initial states was -0.4. All nodes in the graph follow the average consensus protocol given in Equation (2.4) with $\omega = 33$. The state evolution of each node is plotted in Figure 2.4.

Summary

In this chapter, we reviewed some graph terminologies, topological/spectral graph metrics, and information dissemination performance evaluation method. The topological and spectral graph metrics are very important tools to quantify, characterize, and predict the properties of graphs. There is no single metric that can capture the whole graph properties, rather we use various types of graph metrics in the rest of the chapters to evaluate the different properties of network models under our research.

Chapter 3

Network Models

3.1 Introduction

Intuitively, the underlying topology of a network affects its efficiency in exchanging information. For instance, the topology of Internet affects the efficiency of packet routing, that of the wireless sensor networks (WSNs) affects the rate of data fusion, and that of social networks affects the speed of rumor or disease propagation among people [54].

Network topology has to be fast in information exchange, flexible in size, and scalable in information exchange performance over a wide range of sizes. Especially as the networks become larger and denser, the underlying topology plays an even greater role. We, therefore, have been exploring the following questions: (i) is there any specific network model for fast information exchange? (ii) if it exists, is it applicable to real networks such as ad hoc networks? and (iii) what are the issues and solutions?

Among many studies on network models, Watts and Strogatz brought insights into key aspects of efficient network topology. In their seminal work [10], they observed that one can transform a network topology from one with extremely slow information dissemination performance to a very fast one by adding a small amount of randomness to the network, where the resulting networks are known as *Small World Networks* (SWNs). To produce randomness in a network, the authors randomly rewired a small portion of edges of k-regular ring lattices which are known as one of the slowest information dissemination graphs [10]. They evaluated the information dissemination performance of the resulting SWNs using an epidemic model which resembles a distributed networked system. Their experiments revealed that the infectious diseases spread much faster in SWNs than in the original k-regular ring lattices.

More recently in [11], Olfati-Saber utilized the average consensus protocol that solves an average consensus problem in a distributed manner to show the superior information convergence performance of SWNs. Again, in [12], the author investigated the connection between fast mixing Markov chains and quasi Ramanujan graphs (Ramanujan graphs generated by a randomized algorithm). We observed that the graphs displaying fast information dissemination in the aforementioned articles share a common aspect: the randomness in topology.

In this chapter, we will show that BCGs exhibit the fastest information dissemination performance among several different graph families for network sizes ranging from a few hundreds to several thousands nodes. Moreover, topological and spectral graph metrics such as diameter, average path length, and algebraic connectivity confirmed that BCGs can be very efficient in message routing too. To identify the best performing graph family in terms of information dissemination and scalability we also followed the groundwork laid by [11] and [12]. More specifically, we evaluated the information dissemination performance of a graph using the average consensus protocol. We also used graph spectral analysis to compare the performance of BCGs with other graph families such as regular ring lattices, Erdös-Rényi's random graphs, Watts-Strogatz's small world networks, toroidal meshes, and diagonal meshes.

3.2 Regular Graphs

3.2.1 Ring lattices

The k-regular ring lattice is a graph with n vertices arranged on the circle such that each vertex is connected to its k closest neighbors. We implemented the algorithm depicted in Figure 3.1 to generate an adjacency matrix for k-regular ring lattice. Realization of the $2, 4, 8, 16$-regular ring lattices are shown in Figure 3.2.

The maximum distance from a node in a k-regular ring lattice to its immediate neighbors is upper bounded by $\lceil k/2 \rceil$ hops. This distance bound prevents nodes of k-regular ring lattices from having long distance edges connecting it to distant nodes. More importantly in the rest of this chapter, we will discuss how the local connections degrade the information dissemination performance over the network.

```
Require: $n :=$ number of nodes
Require: $k :=$ degree of the lattice
Require: $A := n \times n$ zero matrix
 1: procedure RegRingLatticeGen($n, k, A$)
 2:     for all $u < v \in V(G)$ do            ▷ $u$ and $v$ are integer in $[0, n-1]$
 3:         $m \leftarrow 1$
 4:         while $m \leq k$ do
 5:             if $v = \{(u+n+m)(\textbf{mod } n)\} \wedge \{(u+n-m)(\textbf{mod } n)\}$ then
 6:                 $A[u, v] \leftarrow 1$
 7:                 $A[v, u] \leftarrow 1$
 8:             end if
 9:             $m \leftarrow m + 1$
10:         end while
11:     end for
12: end procedure
```

Figure 3.1: Algorithm for k-regular ring lattice generator.

3.2.2 Meshes

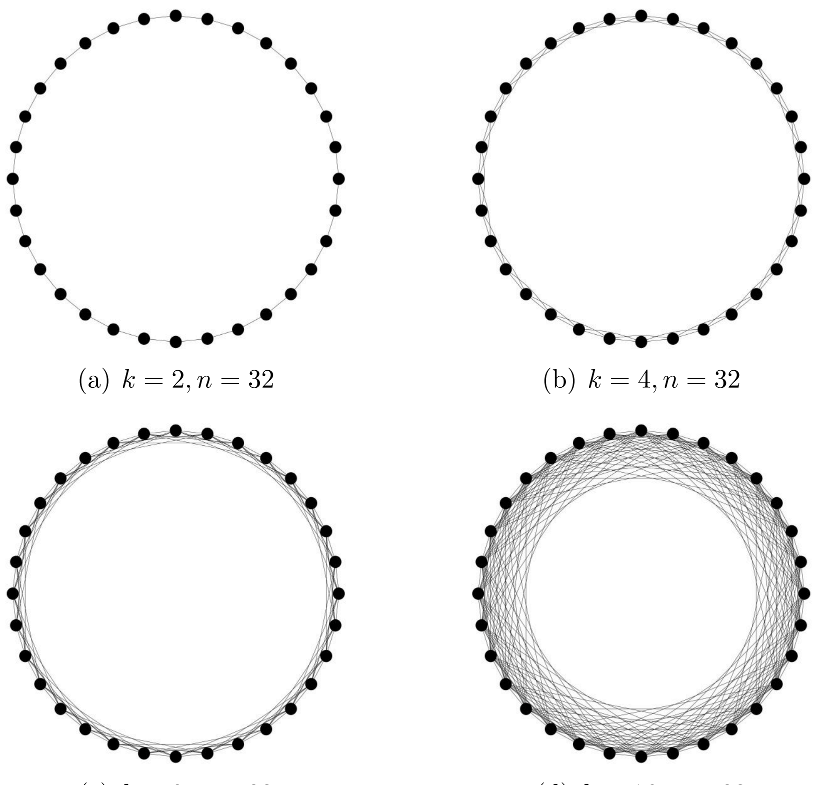
Toroidal mesh. The toroidal mesh in Figure 3.3(a) is a 4-regular, symmetric graph in which nodes are interconnected both in horizontal and vertical directions [55]. Let G_t be a toroidal mesh with $n = l \times k$ nodes where l and k are odd integers. Then, any node located at (x, y) on a Cartesian coordinate plane is connected to the following nodes located at the following four positions:

$$\begin{aligned}
&(x,\ \langle y+1\rangle_l)\ ,\\
&(x,\ \langle y-1\rangle_l)\ ,\\
&(\langle x+1\rangle_k,\ y)\ ,\\
&(\langle x-1\rangle_k,\ y)\ ,
\end{aligned}$$

where $x \in \{-\frac{k-1}{2}, ..., \frac{k-1}{2}\}$, $y \in \{-\frac{l-1}{2}, ..., \frac{l-1}{2}\}$, and the operator

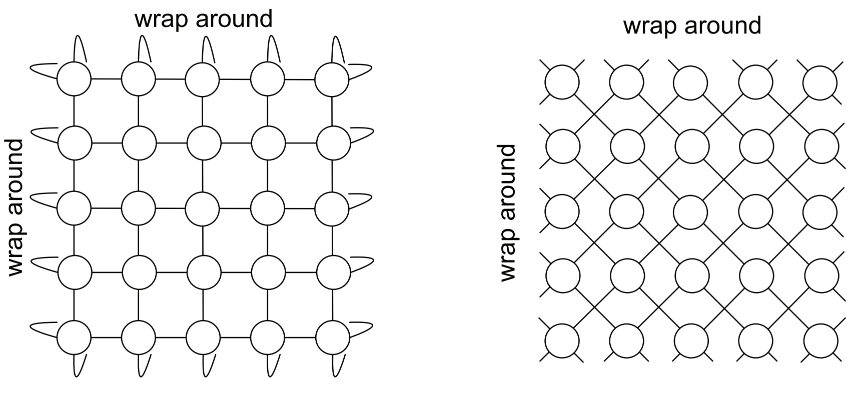
$$\langle x\rangle_k = \begin{cases} x & \text{if } |x| \leq \frac{k-1}{2}\ , \\ x-k & \text{if } x > \frac{k-1}{2}\ , \\ x+k & \text{if } x < -\frac{k-1}{2}\ . \end{cases} \tag{3.1}$$

Tang and Padubidri showed that the diameter of toroidal mesh with $n = l \times k$ nodes, $\text{diam}(G_t) = \frac{l-1}{2} + \frac{k-1}{2}$.


(a) $k = 2, n = 32$ (b) $k = 4, n = 32$

(c) $k = 8, n = 32$ (d) $k = 16, n = 32$

Figure 3.2: Realization of k-regular ring lattice with n=*32* and k=2, 4, 8, and 16.



(a) Toroidal mesh (b) Diagonal mesh

Figure 3.3: 4-Regular toroidal mesh and diagonal mesh.

Diagonal mesh. The diagonal mesh depicted in Figure 3.3(b) is similar to toroidal mesh except that nodes are connected in diagonal directions [55]. Let G_d be a diagonal mesh with $n = l \times k$ nodes where $l \leq k$ are odd integers and let (x, y) be the position of a node in a Cartesian coordinate plane. Then, a node at (x, y) is connected to the nodes at the following four positions:

$$\begin{aligned}
&(\langle x+1\rangle_k,\ \langle y+1\rangle_l),\\
&(\langle x+1\rangle_k,\ \langle y-1\rangle_l),\\
&(\langle x-1\rangle_k,\ \langle y+1\rangle_l),\\
&(\langle x-1\rangle_k,\ \langle y-1\rangle_l),
\end{aligned}$$

where $x \in \{-\frac{k-1}{2}, ..., \frac{k-1}{2}\}$, $y \in \{-\frac{l-1}{2}, ..., \frac{l-1}{2}\}$, and $\langle x\rangle_k$ is the same operator in Eq. (3.1). The authors in [55] also proved that a diameter of diagonal mesh with order $n = l \times k$,

$$\mathrm{diam}(G_d) = \begin{cases} \max(l, \frac{k-1}{2}) & \text{if } l < k\ , \\ n-1 & \text{if } l = k\ . \end{cases} \tag{3.2}$$

3.3 Random Graphs

3.3.1 Erdös-Rényi's random graphs

Erdös and Rényi provided the theoretical foundations of random graphs in [43]. In this section, we briefly review the properties of random graphs and also an algorithm to generate random graphs. Let $\mathcal{G}(n, p)$ be the set of all Erdös-Rényi's random graphs with n nodes and edge probability p. Then, each node of a random graph $G \in \mathcal{G}(n, p)$ is connected to the rest of $(n - 1)$ nodes with the independent edge probability p. The probability for a node to have degree k is given by a binomial distribution

$$\Pr\left[d(v) = k\right] = \binom{n-1}{k} p^k\ (1-p)^{n-1-k}\,. \tag{3.3}$$

Since there are $n(n-1)/2$ node pairs in a graph, the expected number of edges in a random graph,

$$\overline{E} = p\,\frac{n(n-1)}{2}. \tag{3.4}$$

The average degree of a random graph $\overline{d} = 2\overline{E}/n$. Thus, the edge probability $p = \overline{d}/(n-1) \approx \overline{d}/n$ for sufficiently large n. This implies the tight connection between the edge probability and the average degree of a random graph.

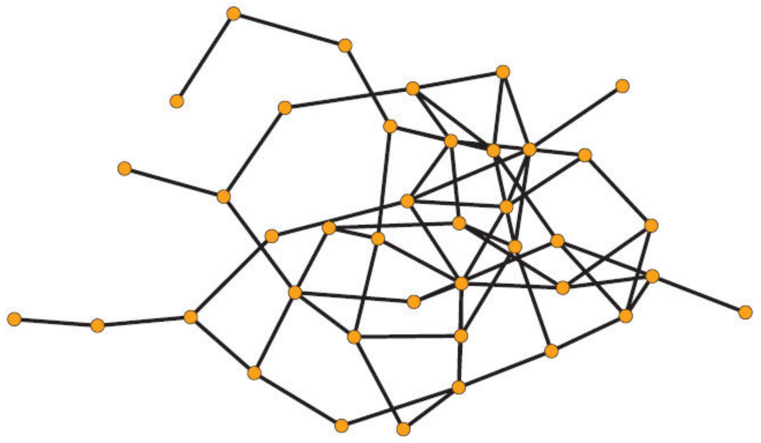


Figure 3.4: Realization of a random graph of 40 vertices and mean degree 3.

We generate a random graph by drawing a uniformly distributed random number $r \in [0, 1)$ for each possible pair of vertices and join them with an edge if $r < p$. Thus, each element of the adjacency matrix of a random graph is defined by

$$a_{u,v} = a_{v,u} = \begin{cases} 1 & \text{if } r < p, \\ 0 & \text{otherwise,} \end{cases}$$

for all $u < v \in V(G)$. This yields an $n \times n$ adjacency matrix for a random graph $G \in \mathcal{G}(n, p)$. The number of trials in generating a $n \times n$ adjacency matrix of random graph is in $O(n^2)$ regardless of the mean number of edges to be generated. It is obvious that most of trials for joining vertex pairs will fail for small edge probability p as is the case for large sparse random graphs. To avoid speed bottleneck in generating a large sparse random graph, we used the efficient random graph generating algorithm proposed in [56]. Figure 3.5 summarizes the random graph generator algorithm. The realization of a random graph with 40 vertices and mean degree of 3 is depicted in Figure 3.4.

Note that due to the way we generate random graphs, it is possible to produce disconnected graphs. We discard these disconnected graphs because multiple connected components prevents the graph from reaching the graph wide consensus. The connectivity test can be done by counting the zero degree vertices or zero diagonal elements of the Laplacian matrix. Since the number of the zeros in the eigenvalues of a graph Laplacian represents the number of connected components in a graph, we simply compute the algebraic connectivity of the graph. If the algebraic connectivity is zero, then the graph is disconnected.

```
Require: $n :=$ number of vertices
Require: $p = (0, 1)$                                        ▷ edge probability
Require: $A := n \times n$ zero matrix                       ▷ adjacency matrix
 1: procedure RandGraphGen($n, p, A$)
 2:     $u \leftarrow 1$                                     ▷ $u :=$ integer label of node
 3:     $v \leftarrow -1$                                    ▷ $v :=$ integer label of node
 4:     while $u < n$ do
 5:         choose $r \in [0, 1)$ uniformly at random
 6:         $v \leftarrow v + 1 + \left\lfloor \frac{\log(1-r)}{\log(1-p)} \right\rfloor$
 7:         while $u \leq v \vee u < n$ do
 8:             $v \leftarrow v - u$
 9:             $u \leftarrow u + 1$
10:         end while
11:         if $u \leq n$ then
12:             $A[u, v] \leftarrow 1$
13:             $A[v, u] \leftarrow 1$
14:         end if
15:     end while
16: end procedure
```

Figure 3.5: Algorithm for random graph generator [56].

3.3.2 Watts-Strogatz's small world networks

Small world networks are one category of network models with a set of nodes connected to mostly neighboring nodes (highly clustered) but with a few long range connections that provide shortcuts over clusters (short average path length). In [10], Watts and Strogatz experimented on small-world phenomenon by randomly rewiring the edges of regular ring lattice with a small rewiring probability. The results of their experiments showed that the average path length drops dramatically when *small-world phenomenon* happens, but at the same time the clustering coefficient representing the magnitude of local clustering remains almost in the same range.

Later in [11], Olfati-Saber further verified Watts and Strogatz's observations in a more quantitative manner by introducing a distributed decision making protocol called the *average-consensus protocol.* In [11], the author showed that it is possible to achieve ultrafast information dissemination (or network-wide consensus) by constructing a favorable network topology. For example, applying the *random-rewiring* scheme on regular ring lattice increases the algebraic connectivity by 1000 folds. Similarly, in [12], the Ramanujan graphs, a class of regular Cayley graphs, were proven to be an efficient network topology for ultrafast information fusion.

3.4 Cayley Graphs

In the previous sections, we reviewed regular and random graphs. A Cayley graph is a special family of pseudo-random and regular graphs constructed from a finite group of which elements correspond to the nodes of the graph [57–59]. Connections between nodes of Cayley graphs are defined by a group operator and a set of generators. The formal definition of Cayley graphs is as follows.

Definition 3.1 (Cayley graph [59]). Let V be a group and let $\mathbb{G}$ be a non-empty generating set. The Cayley graph $\Gamma = \mathit{\Gamma}(V, \mathbb{G})$ is a directed graph with vertices from the group V and with directed edges $e_{v,u}$ if $u = v * g$ for some $g \in \mathbb{G}$ and $u \neq v \in V$, where $*$ is the group operation.

The definition of Cayley graphs requires vertices to be elements of a group but does not specify a particular group. Thus, a Cayley graph can be generated over an arbitrary finite group, so there can be many varieties of Cayley graphs.

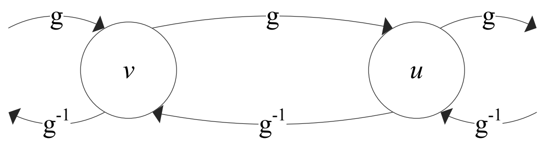


Figure 3.6: Connection rule of Borel Cayley Graph.

3.5 Borel Cayley Graphs

The Borel Cayley graphs (BCGs) are Cayley graphs constructed from Borel subgroups. The BCGs are regular, vertex transitive, and pseudo-random graphs [58]. The definition of a Borel subgroup is given below.

Definition 3.2 (Borel subgroup [58]). Let $V \in BL_2(Z_p)$ be a Borel subgroup of the nonsingular upper triangular 2×2 matrices $GL_2(Z_p)$ with a parameter a such that $a \in Z_p \setminus \{0, 1\}$, then

$$V = \left\{ \begin{pmatrix} x & y \\ 0 & 1 \end{pmatrix} : x = a^t (\textbf{mod}\ p),\ y \in Z_p,\ t \in Z_k \right\}, \tag{3.5}$$

where p is prime, and k is the smallest positive integer such that $a^k = 1(\mathbf{mod}\ p)$.

From the definition of a Borel subgroup, we define the Borel Cayley graph as follows:

Definition 3.3 (Borel Cayley graph [58])**.** Let V be a Borel subgroup and let $\mathbb{G}$ be a generating set such that $\mathbb{G} \subseteq V \setminus \{I\}$, then $\mathbf{C} = \mathcal{C}(V, \mathbb{G})$ is a Borel Cayley graph with vertices represented in 2×2 matrix elements of V and with directed edge from v to u if $u = v * g$, where $u \neq v \in V$, $g \in \mathbb{G}$, and $*$ is a modulo-p matrix multiplication chosen as the group operation.

Note that, in the definition of Borel Cayley graph, we explicitly exclude the identity element from the generating set to avoid self-loops in the graph. In the following sections, we discuss key properties of BCGs.

3.5.1 Node representation

Nodes in BCGs are defined in the 2×2 matrix domain. In earlier work, Arden and Tang [57] showed that all Cayley graphs can be represented as generalized chordal rings (GCRs) where their nodes are grouped into $q = k$ classes according to the modulo-q operation. Specifically, there is a function that maps the Borel subgroup represented in 2×2 matrix elements onto the non-negative integer domain $\mathbb{N}_0$ of GCR. That is, each node represented by a Borel subgroup element is mapped onto a number between 0 and $n-1$, where n is the number of elements in the Borel subgroup:

$$\begin{aligned} f: \quad V \quad &\rightarrow \quad \mathbb{N}_0, \\ \begin{pmatrix} a^t & y \\ 0 & 1 \end{pmatrix} &\mapsto \quad yq + t. \end{aligned} \tag{3.6}$$

The systematic representation of BCGs in the GCR integer domain makes BCGs tractable for routing and other networking issues [23, 57].

3.5.2 Connection rule

In Definition 3.3, BCG's edges between nodes are algebraically established by the connection rule of Cayley graphs. In Figure 3.6, for example, any two nodes u and v are connected by two directed edges labeled with generators g and its inverse g^{-1}. The modulo-p multiplication between a node $v \in V$ and the generator $g \in \mathbb{G}$ yields another 2×2 matrix $u \in V$. Since the relationship between the nodes u and v satisfies the BCG's connection rule ($u = v * g$),

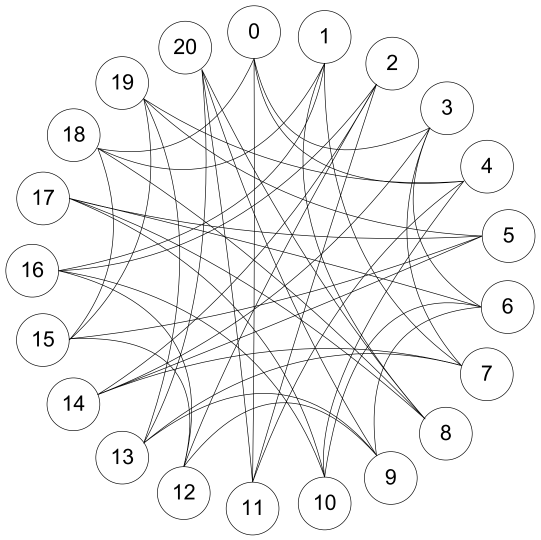


Figure 3.7: A 21-node, 4-regular BCG.

there is a directed edge, $e_{v,u}$, from node v to u. Similarly, there exists another directed edge in the backward direction, $e_{u,v}$, that satisfies $v = u * g^{-1}$. Notice that a generator g and its inverse g^{-1} in conjunction establishe an undirected (bidirectional) edge between nodes u and v. Because we are only considering undirected BCGs in this paper, we assume that a generating set $\mathbb{G}$ is closed under inversion, i.e., $g \in \mathbb{G} \Leftrightarrow g^{-1} \in \mathbb{G}$, for all g, $g^{-1} \in \mathbb{G}$. Furthermore, the nodal degree of BCGs is determined by the order of the generating set. Since each pair of generators contributes degree-2 for each node, two pairs of generators $\{g_1, g_1^{-1}\}$ and $\{g_2, g_2^{-1}\}$, for example, generate undirected 4-regular BCGs.

3.5.3 Examples

Figure 3.7 shows an example of a 21-node, 4-regular BCG with parameters $p = 7$, $k = 3$, $a = 2$, and generating set $\mathbb{G} = \{g_1 = \left(\begin{smallmatrix}1 & 1\\ 0 & 1\end{smallmatrix}\right), g_2 = \left(\begin{smallmatrix}2 & 1\\ 0 & 1\end{smallmatrix}\right), g_1^{-1}, g_2^{-1}\}$. From the figure, we can observe that the connections look random even though the graphs is deterministically constructed from the given BCG parameters. Note that a BCG looks the same from any node (vertex transitive) which simplifies routing [23]. In Section 3.7, we will show how this pseudo-random property of BCGs contributes to the ultrafast information dissemination or data exchange over a network.

3.6 How to Select BCG Parameters?

This section provides a heuristic guideline for selecting BCG generators that produce connected BCGs. The definition of Borel Cayley graphs explains how the connections of a BCG are governed by the choice of generators. In fact, we observed that most generator pairs yield densely connected BCGs with small diameters. However, it is also observed that some generator pairs generate disconnected BCGs. In general, a disconnected graph is undesirable. For example, networked systems such as computer networks, wireless communication networks, and cooperative vehicles require the underlying network topology to be connected to ensure communication between any pair of nodes in the network. Thus, it is critical to avoid generating disconnected graphs by choosing appropriate parameters for the generators.

3.6.1 BCG samples

We generated a set of 4-regular BCG samples associated with different generator pairs to identify a pattern of *bad* generator pairs that generate disconnected BCGs. We consider generators $g_1 = \left(\begin{smallmatrix} a^{t_1} & 0 \\ 0 & 1 \end{smallmatrix}\right)$, $g_2 = \left(\begin{smallmatrix} a^{t_2} & 1 \\ 0 & 1 \end{smallmatrix}\right)$ and their inverses g_1^{-1}, g_2^{-1}. In the rest of this section, we use (t_1, t_2) to represent the generator pairs. We set the range of t_1 and t_2 to $1 \leq t_1 < t_2 \leq 10$ instead of using all (t_1, t_2) pairs in the range $0 \leq t_1 < t_2 \leq (k-1)$ to limit the sample size. Thus, the total number of generator combinations is 45 where each (t_1, t_2) pair defines a generator set that generates a unique BCG with different connectivity pattern.

Table 3.1 summarizes the parameters of BCGs used in our experiments. The parameters p, k, and a have been chosen based on the definition of a Borel subgroup (see Definition 3.2). We used two scenarios for generating BCGs: Scenario A generates BCGs with sizes $n = 1081$, 2265, 3081, 4112 and 5253, while Scenario B generates BCGs with $n = 2211$ and 4063. We use the data from Scenario A to find patterns of *good* and *bad* generator pairs. On the other hand, Scenario B is designed to validate our observations. For each of the generated BCG samples, we computed its algebraic connectivity.

3.6.2 Patterns of bad generators

We focus on finding patterns from the bad generator pairs that construct disconnected graphs. Based on 45 (t_1, t_2) pairs and their corresponding BCGs with sizes $n = 1081$, 2265, 3081, 4112, and 5253, we found a pattern in *good* and *bad* generators. The algebraic connectivity of the generated graphs reveals that most of the BCGs generated were connected graphs while a smaller

Table 3.1: BCG sizes and corresponding parameters.

scenario	n	p	k	a	samples
A	1081	47	23	2	45
	2265	151	15	2	45
	3081	79	39	2	45
	4112	257	16	2	45
	5253	103	51	2	45
B	2211	67	33	6	45
	4063	239	17	6	45

portion was disconnected. In fact, $4, 3, 10$ and 4 samples out of the 45 BCG samples for $n = 2265$, 3081, 4112 and 5253, respectively, were disconnected graphs.

In Table 3.2, we summarize the parameters of the disconnected BCGs, p, k and (t_1, t_2). In the table, $\mathbf{d}(k)$ and $\mathbf{cd}(t_1, t_2)$ are the set of k's divisors and the set of the common divisors of t_1 and t_2[1], respectively. Note that if a BCG is disconnected, one of k's divisors always appears in the common divisors of (t_1, t_2) pairs. Also, the opposite was always true. That is, if a BCG is disconnected, then $\mathbf{cd}(t_1, t_2) \subset \mathbf{d}(k)$. In fact, 100% of the generator pairs in Table 3.2 share common divisors with the divisors of k. For example in the 2265-node case, the common divisors producing disconnected graphs $\mathbf{cd}(t_1, t_2) = \{3, 5\}$ are also the divisors of $k = 15$. In addition, the common divisors of all the other (t_1, t_2) pairs generating connected graphs other than $(3, 6)$, $(3, 9)$ and $(6, 9)$ are not the factors of k.

Furthermore, we also conjecture that if the BCG is connected, then $\mathbf{cd}(t_1, t_2)$ is not a factor of k. For example, the parameter k of 1081-node BCGs is the prime number 23 of which only two factors are the trivial factor 1 and itself. The fact that the divisor 23 of k is a prime number and that it is not within the range of t_1 or t_2 guarantees that no common divisor $\mathbf{cd}(t_1, t_2)$ divides k.

To support this conjecture, we further experimented with Scenario B in Table 3.1. We found that, in Table 3.3, the (t_1, t_2) pairs of all disconnected 2211-node Borel Cayley graphs are the same as the ones of disconnected 3081- and 5253-node Borel Cayley graphs. Our conjecture is further supported since the common divisors of disconnected graphs for all 2211, 3081 and 5253 cases are 3 which is also a divisor of k. Moreover, we observed that there is no disconnected graph generated for 4063-node Borel Cayley graphs in the range

[1] Of course, we exclude trivial divisors 1 and parameter itself. For $\mathbf{d}(k)$, however, if the parameter k itself is the only divisor, then we use k as a divisor of k.

Table 3.2: (t_1, t_2) generating disconnected BCGs (Scenario A).

n	p	k	$\mathbf{d}(k)$	(t_1, t_2)	$\mathbf{cd}(t_1, t_2)$
1081	47	23	23	-	-
2265	151	15	3, 5	(3,6), (3,9)	3
				(5,10)	5
				(6,9)	3
3081	79	39	3, 13	(3,6), (3,9)	3
				(6,9)	3
4112	257	16	2	(2,4), (2,6), (2,8), (2,10)	2
				(4,6), (4,8), (4,10)	2
				(6,8), (6,10)	2
				(8,10)	2
5253	103	51	3, 17	(3,6), (3,9)	3
				(6,9)	3

Table 3.3: Generator pairs generating disconnected BCGs (Scenario B).

n	p	k	$\mathbf{d}(k)$	(t_1, t_2)	$\mathbf{cd}(t_1, t_2)$
2211	67	33	3, 11	(3,6), (3,9)	3
				(6,9)	3
4063	239	17	17	-	-

of (t_1, t_2) pairs we considered. That is because the sole factor $\mathbf{d}(k) = 17$ for 4063-node BCGs does not fall into the experimented range $1 \leq t_1 < t_2 \leq 10$ and hence, there is no generator pairs of which common divisors divide the factor of k.

3.6.3 Guideline

Based on our observations and conjectures in the previous section, we suggest the following guidelines for choosing k and (t_1, t_2) pairs to generate a connected BCG.

Odd k: If k is even, then the divisor $\mathbf{d}(k) = 2$ is a common divisor of all $(t_1, t_2) = (even, even)$ pairs which will generate disconnected BCGs. So, by choosing an odd k, we avoid cases generating disconnected BCGs.

Prime k: If possible, choose a prime k.

Given k, choose (t_1, t_2) such that $\mathbf{cd}(t_1, t_2) \not\subset \mathbf{d}(k)$: Once k is chosen, the generator (t_1, t_2) has to be chosen such that any common divisor of t_1 and t_2 ($\mathbf{cd}(t_1, t_2)$) is not a divisor of k denoted as $\mathbf{d}(k)$.

3.7 Information Dissemination Performance of Network Models

3.7.1 Setup

We evaluate the convergence speed of the average consensus protocol over the benchmark graph families for a wide range of sizes and degrees. The benchmark graphs evaluated include BCGs, random graphs, small world networks, regular ring lattices, toroidal meshes, and diagonal meshes. The parameters used for generating the benchmark graphs are summarized in Table 3.5.

We did not consider disconnected graphs in our evaluation. For the random graphs, we collected and evaluated 30 connected graphs. Small world networks were generated following [10] with the rewiring probabilities of 0.01, 0.1, and 0.2. We report results averaged over 10 small world networks. Similarly, results for each BCG of a given size and degree were averaged over 5 BCGs constructed with different generator matrices. On the other hand, for the regular graphs such as toroidal mesh, diagonal mesh, and ring lattices, it was not required to average the performance metrics over several instances of the graphs.

Table 3.4: Consensus protocol parameters.

Parameter	**Description and Assignment**
consensus protocol	average-consensus protocol (Figure 3.8)
initial state	uniformly distributed random integers in $[-5, 5]$
step-size	$\omega = 2d_{max}(G) + 1$
precision	0.001

To guarantee asymptotic convergence of the average consensus protocol, we set the step-size to the maximum allowed value [53]: $\omega = (2d_{max}(G)+1)$. Each node in the graph was initialized with an integer random number between -5 and 5. The average consensus protocol reaches an agreement once the state value of every node equals the average of all initial values within a precision of 0.001. That is, we declare the network-wide consensus if $\mathbf{r} = c\mathbf{1}$, where $\mathbf{r}$ is a column vector containing the state value of every node, $\mathbf{1}$ is all ones column vector, and c is the average of all initial state values. Table 3.4 summarizes the consensus protocol parameters used in our simulations. Our simulations are performed on the OpenSUSE 11.1 (x86-64) operating system and Linear Algebra PACKage (LAPACK) [60] is incorporated to compute the eigenvalues of the graph Laplacian.

3.7.2 Performance metrics

To characterize the information dissemination performance of a graph, we used the following metrics:

- **Algebraic connectivity:** See Definition 2.16.
- **Spectral radius / algebraic connectivity ratio:** See Definition 2.16 and Definition 2.17.
- **Consensus steps:** The number of steps (iterations) required for all nodes in a network (or a graph) to reach the global agreement (i.e., consensus value) using the average consensus protocol described in Section 3.8.

3.7.3 Results

In this section, we compare the convergence speeds of various families of graphs for different sizes and degrees.

```
Require: $L := n \times n$ Laplacian matrix of $G$
Require: $\mathbf{r}_{init}$ := column vector of all initial states
Require: $c = \frac{1}{n}\sum_i r_i, \forall r_i \in \mathbf{r}_{init}$
1: procedure AVERAGECONSENSUSPROTOCOL($L, \mathbf{r}_{init}$)
2:     $\tau \leftarrow 0$
3:     $\mathbf{r} \leftarrow \mathbf{r}_{init}$
4:     $\omega \leftarrow 2d_{max}(G) + 1$
5:     repeat
6:         $\mathbf{r} \leftarrow (I - \frac{1}{\omega}L)\,\mathbf{r}$
7:         $\tau \leftarrow \tau + 1$
8:     until $\mathbf{r} = c\mathbf{1}$                  ▷ precision set to $10^{-3}$ for simulations
9: end procedure
```

Figure 3.8: Algorithm for average consensus protocol.

Table 3.5: Graph families and parameters.

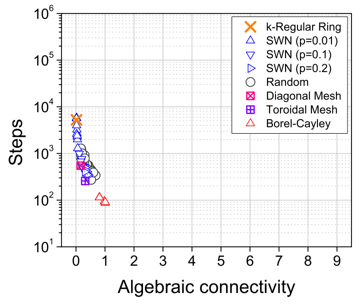
graph family	size	degree	Number of samples
Random graph[†]	110, 171, 253	4, 8, 16	30
	1081, 2000, 3000,4000, 5000	8, 16	20
Small-world network[†]	110, 171, 253, 1081, 2000, 3000, 4000, 5000	4, 8, 16	10
Regular ring lattice	110, 171, 253, 1081, 2000, 3000, 4000	4, 8, 16	1
Borel Cayley graph	110, 171, 253, 1081	4, 8, 16	5
	2162, 3403, 4970	4, 8, 16	1
Toroidal mesh	110, 171, 253, 1081, 2000, 3000, 4000, 5000	4	1
Diagonal mesh	110, 171, 253, 1081, 2000, 3000, 4000, 5000	4	1

[†] Non-regular graphs, thus the degrees are mean values.

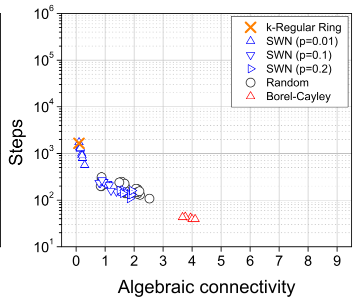
In Figure 3.9, we plot the number of steps required to reach the network-wide consensus as a function of the algebraic connectivity for different families of graphs over a range of graph sizes and degrees. As shown in the figure, in all cases, BCGs consistently exhibit better information dissemination performance than other benchmark graphs. Furthermore, we noticed that the information dissemination performance improves as degree increases. Figure 3.9 also confirms the known result that the larger the algebraic connectivity, the faster consensus protocol convergence speed. Similar results were observed for graphs of size ranging from 2000 to 5000 nodes and are not repeated here.

Figure 3.10 plots the number of steps to reach a consensus versus the ratio of spectral radius over algebraic connectivity (λ_n/λ_2). It has been known in the literature that such a graph spectral property ratio is a good performance indicator for the information diffusion rate [45]. The results shown in Figure 3.10 are consistent with those in [45]. The performance of the BCGs have the smallest ratio and fastest convergence speed comparing to the benchmark graphs with the same sizes and degrees.

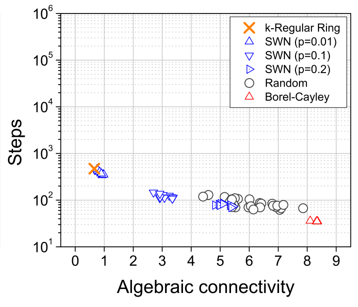
Figure 3.11 demonstrates how the information dissemination performance scales over different network sizes across different graph families. Comparison among different graph families with the same degree graphs shows that BCGs consistently achieve the fastest information dissemination speed. Furthermore, the BCGs scale very well over network sizes ranging from 100 to 5000 nodes. The random graphs and the Small World Networks (SWNs) with reasonably high rewiring probability ($p = 0.1$, and 0.2) scales relatively well. On the contrary, the regular ring lattices scale very poorly over different network sizes. These results confirm that we can improve the information dissemination performance of regular ring lattices by rewiring only a small portion of its local links, as reported in [11]. Moreover, for a given degree, as the size of the graph increases, the performance gap among various graphs families further widens. For a given graph size, the consensus protocol convergence speed over the same graph family improves with increasing degree.



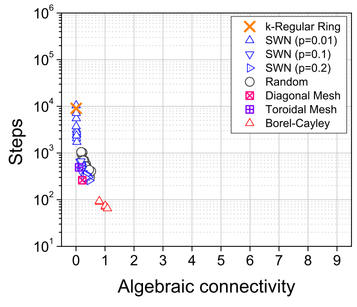
(a) $n = 110, d(G) = 4$



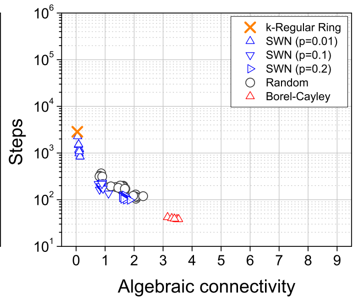
(b) $n = 110, d(G) = 8$



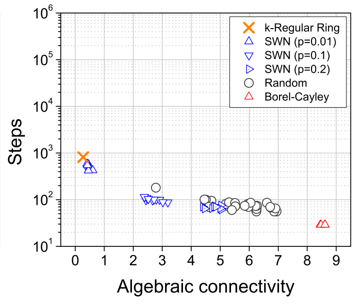
(c) $n = 110, d(G) = 16$



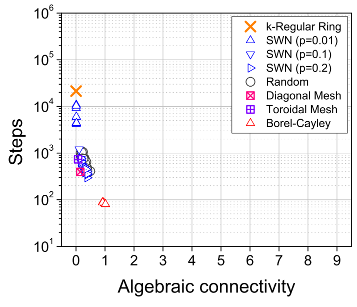
(d) $n = 171, d(G) = 4$



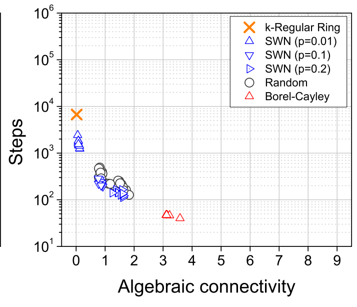
(e) $n = 171, d(G) = 8$



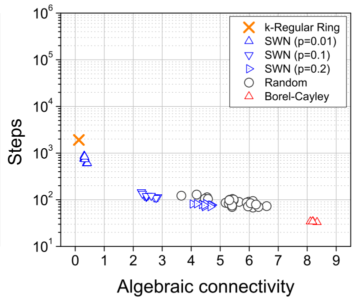
(f) $n = 171, d(G) = 16$



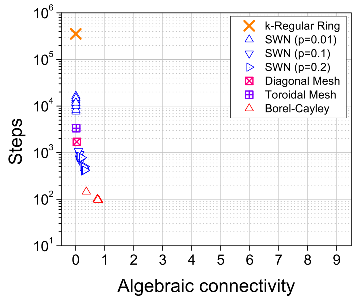
(g) $n = 253, d(G) = 4$



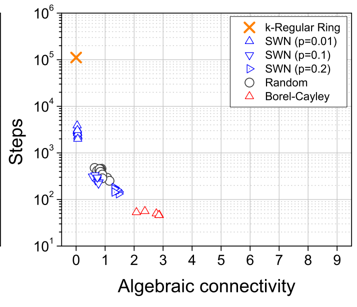
(h) $n = 253, d(G) = 8$



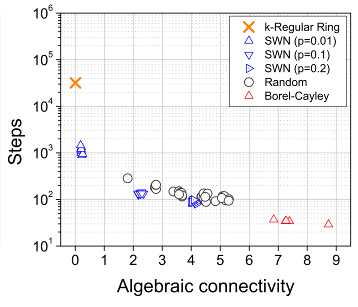
(i) $n = 253, d(G) = 16$



(j) $n = 1081, d(G) = 4$

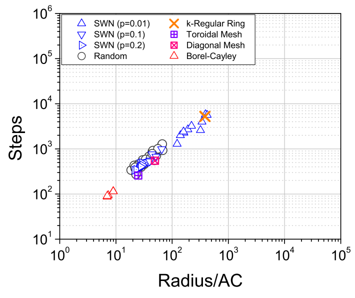


(k) $n = 1081, d(G) = 8$

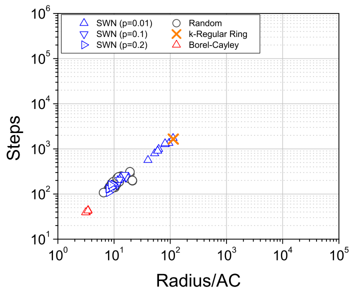


(l) $n = 1081, d(G) = 16$

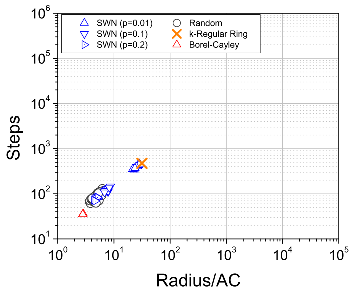
Figure 3.9: Information dissemination performance versus algebraic connectivity (λ_2) for $n = 110$, 171, 253, and 1081.



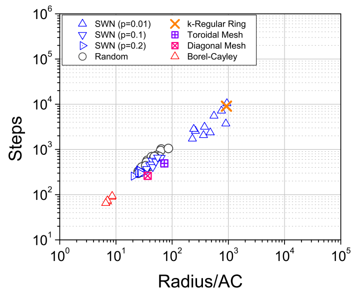
(a) $n = 110, d(G) = 4$



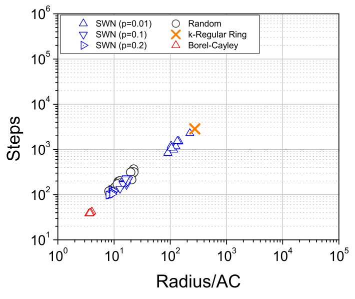
(b) $n = 110, d(G) = 8$



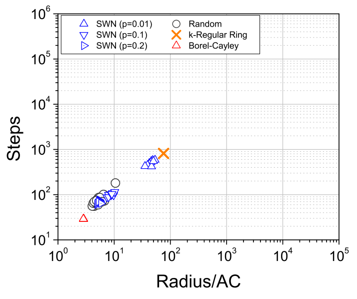
(c) $n = 110, d(G) = 16$



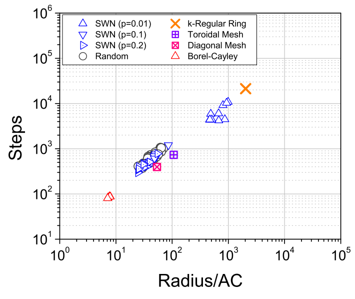
(d) $n = 171, d(G) = 4$



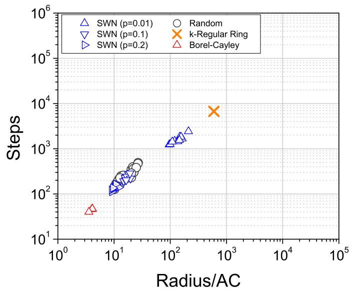
(e) $n = 171, d(G) = 8$



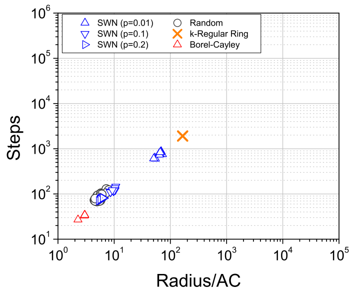
(f) $n = 171, d(G) = 16$



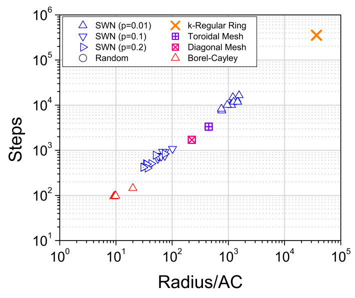
(g) $n = 253, d(G) = 4$



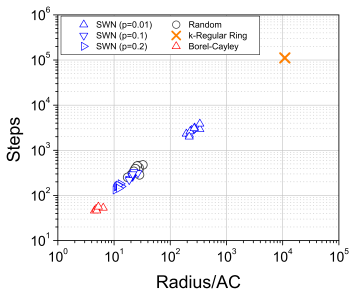
(h) $n = 253, d(G) = 8$



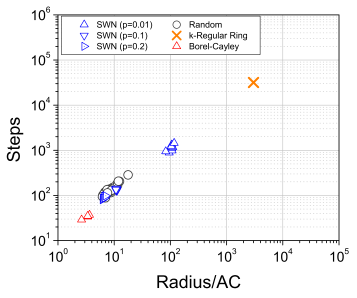
(i) $n = 253, d(G) = 16$



(j) $n = 1081, d(G) = 4$

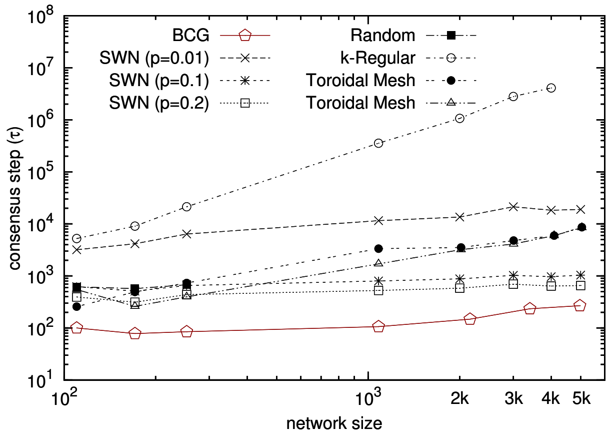


(k) $n = 1081, d(G) = 8$

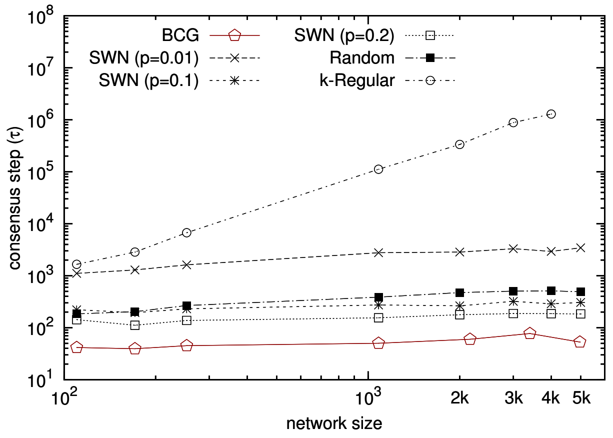


(l) $n = 1081, d(G) = 16$

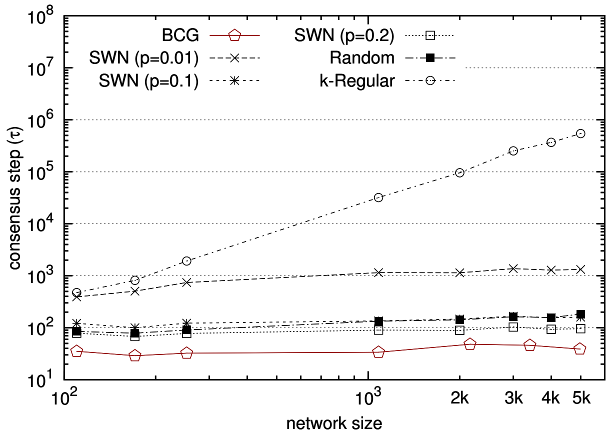
Figure 3.10: Information dissemination performance versus the ratio of spectral radio to algebraic connectivity (λ_n/λ_2) for $n = 110$, 171, 253, and 1081.



(a) $d(G) = 4$



(b) $d(G) = 8$



(c) $d(G) = 16$

Figure 3.11: Convergence steps vs. network sizes.

Summary

In this chapter, we reviewed various network models including Erdös-Rényi's random graphs, Watts-Strogatz's small world networks, k-regular ring lattices, toroidal meshes, diagonal meshes, and Borel Cayley Graphs. We evaluated information dissemination speed of each network model using several metrics including average consensus protocol convergence speed, algebraic connectivity, and the ratio of spectral radius to algebraic connectivity.

We found that BCGs are one of the most favorable graphs that have the following properties: (i) the fastest information dissemination speed among considered benchmark graphs, and (ii) the best scalability among the benchmark graphs with the network sizes from $n = 100$ to 5000 nodes. We attribute this superior information dissemination performance and scalability of the BCGs to its pseudo-random connections that help to facilitate even distribution of information exchange among nodes in the network.

Chapter 4

Quasi Borel Cayley Graph

4.1 Introduction

In the previous chapter, we discussed various advantageous properties of BCGs including (a) integer domain representations as GCR (Generalized Chordal Rings), (b) an algebraic graph construction, (c) vertex transitivity, (d) a constant degree, and (e) an ultrafast information dissemination performance. We also found that the information dissemination performance of BCGs scales very well over a wide range of network sizes ranging from 1000 to 5000 nodes.

However, we also recognized that it has been challenging to apply BCGs to real networks because of BCGs' lack of size flexibility. As noted in Definitions 3.2 and 3.3, the size of a BCG is $p \times k$, where p is a prime, and k is a factor of $p-1$. Because of the limited choices in choosing BCG parameters p

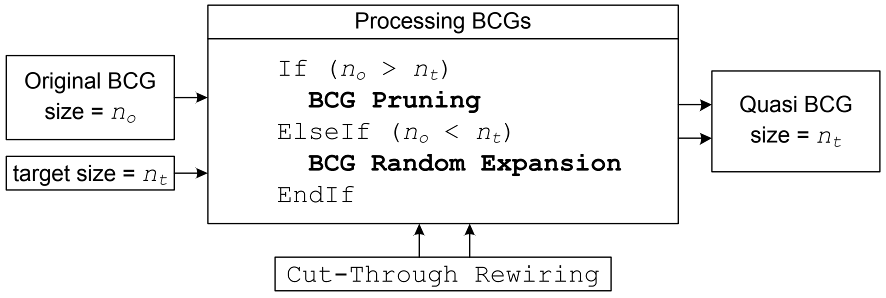


Figure 4.1: The quasi Borel Cayley graph generation process.

Table 4.1: The original sizes n_o of example BCGs and their corresponding BCG parameters p, k, and a.

p	k	a	n_o	p	k	a	n_o	p	k	a	n_o
7	3	2	21	109	36	2	3924	151	45	2	6795
17	16	3	272	257	16	2	4112	337	21	2	7077
47	23	2	1081	97	48	11	4656	223	37	2	8251
53	26	6	1378	521	10	5	5210	331	30	2	9930
67	33	2	2211	103	51	2	5253	107	106	7	11342
79	39	2	3081	241	24	2	5784	769	48	3	36912
83	41	3	3403	199	33	5	6567	223	222	11	49506

and k, it is not possible to construct a BCG of arbitrary size. For example, Table 4.1 shows examples of BCG sizes and corresponding BCG parameters. To facilitate realistic applications of BCGs as network topologies, we propose graph resizing algorithms that produce Quasi BCGs with flexible sizes, while preserving the aforementioned advantageous properties of the original BCGs.

A Quasi BCG is a BCG that is transformed by our *BCG Pruning* algorithm or *BCG Random Expansion* algorithm. Figure 4.1 illustrates a diagram for Quasi BCG generation method that takes the original BCG (with the original size n_o) and target size (n_t) as inputs. Using those two inputs, our algorithms generate Quasi BCGs of target size n_t. Note that in the rest of this section, we are considering undirected 4-regular BCGs constructed by two generators and their inverses. In the following sections, we describe our proposed BCG Pruning and Random Expansion algorithms.

4.2 BCG Pruning

The proposed BCG pruning algorithm randomly selects a node to be removed from the original BCG, prunes the selected node, and rewires broken edges of remaining nodes to maintain connectivity. The node removal and edge rewiring processes are repeated until the desired target size is reached. Figure 4.2 summarizes our BCG Pruning algorithm. In the next section, we present the *Cut-Through Rewiring (CTR)* algorithm which is a key to preserve the deterministic connection rule of BCGs.

```
Require: $V :=$ a vertex set of Borel Cayley Graph
Require: $0 < n_t < n_o$
1: procedure GraphResizing($V, n_t$)
2:     while $n_t < |V|$ do
3:         select a node $v \in V$ uniformly at random
4:         $V \leftarrow V \setminus \{v\}$                ▷ prune node $v$
5:         perform Cut-Through Rewiring                ▷ Figure 4.6
6:     end while
7: end procedure
```

Figure 4.2: Algorithm for BCG pruning.

4.2.1 Cut-through rewiring

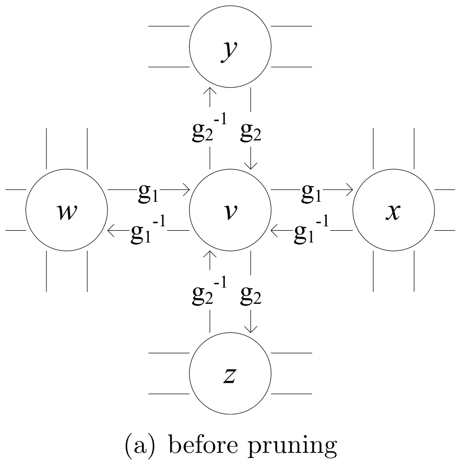
Whenever a node is removed from a BCG, the BCG Pruning algorithm rewires the edges of the remaining nodes using the CTR algorithm to preserve the connectivity of the pruned BCG. To make CTR algorithm deterministic, we exploit the BCG's generators and deterministic connection rule. Recall in Definition 3.3 that a node u is connected to another node v if and only if $u = v * g$ for $g \in \mathbb{G}$. From this connection rule, we define the g-direction as follows:

Definition 4.1 (g-direction)**.** Let g be a generator of a Borel Cayley graph, then the g-direction between two neighboring nodes v and w is the logical direction of the directed edge $e_{v,w}$ such that $w = v*g$. We denote two connected nodes in the g-direction by $v \overset{g}{\hookrightarrow} w$.

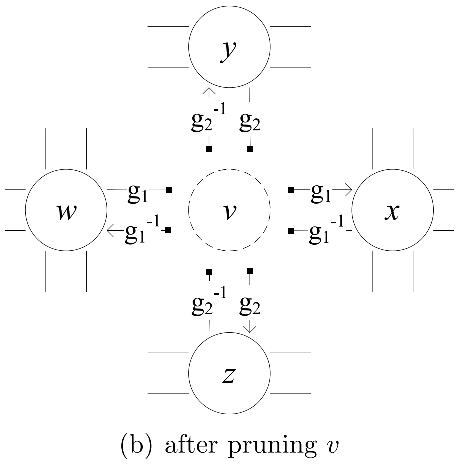
Based on the above definition, let us consider a 4-regular BCG formulated by generators g_1, g_2, g_1^{-1}, and g_2^{-1}. Then, any node $v \in V$ is connected to its four neighbors in the g_1, g_2, g_1^{-1}, and g_2^{-1} directions with the respective neighbors $\mathcal{N}_v = \{x, y, w, z\}$, where $x = v * g_1$, $y = v * g_2$, $w = v * g_1^{-1}$, and $z = v*g_2^{-1}$. Now, suppose we are pruning v in Figure 4.3(a). Then, as shown in Figure 4.3(b), the four neighbors of v, $\mathcal{N}_v = \{w, x, y, z\}$, lose their connections to v.

Our CTR algorithm preserves the connectivity of the graph by establishing edges that tunnel or cut-through the pruned node v in the g_1, g_2, g_1^{-1}, and g_2^{-1} directions as illustrated in Figure 4.3(c). More specifically, after node v is pruned, our rewiring algorithm rewires the node w to the new neighbor x found by the following group multiplication:

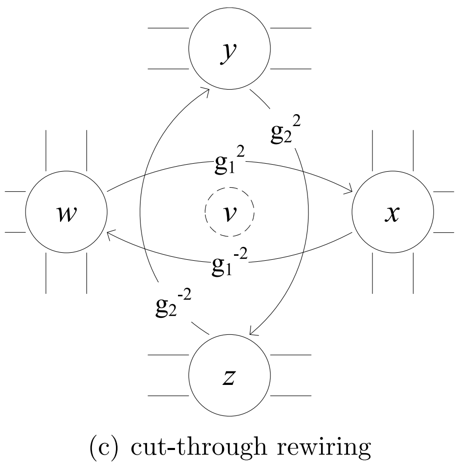
$$w * g_1^2 = \underbrace{w * g_1}_{v} * g_1 = v * g_1 = x \ .$$



(a) before pruning



(b) after pruning v



(c) cut-through rewiring

Figure 4.3: Pruning and Cut-Through Rewiring.

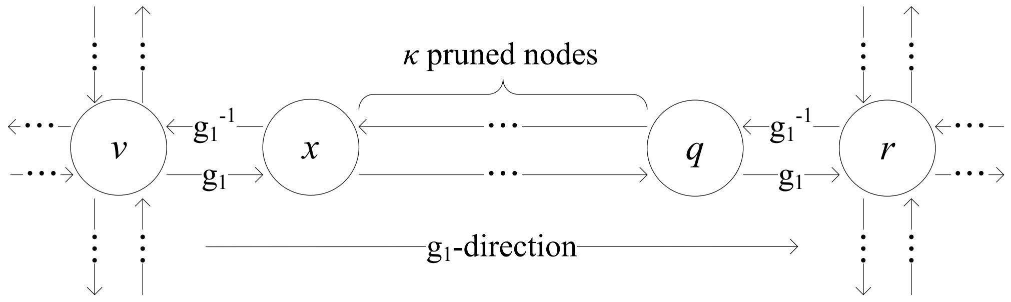
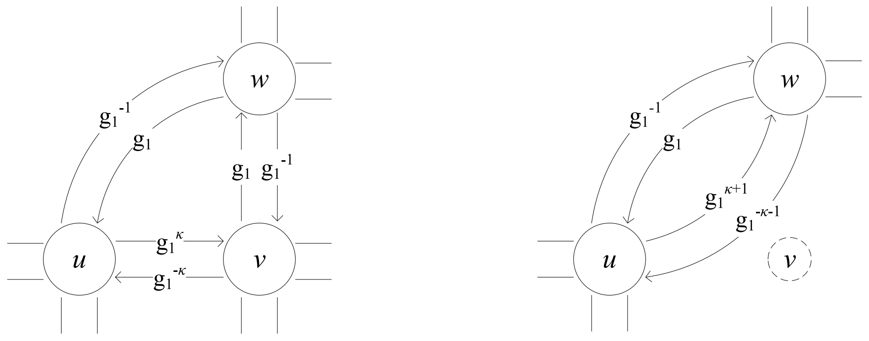


Figure 4.4: Cutting-through multiple edges.



(a) K_3 cycle (b) Multiple edges

Figure 4.5: Handling K_3 cycle.

Similarly, CTR algorithm determines the new neighboring node of x to be $w = x * g_1^{-1} * g_1^{-1} = x * g_1^{-2}$. Note that this simple rewiring operation is applied to the rest of the nodes, y and z, for g_2 and g_2^{-1} directions as well.

We also note that it is necessary to cut-through multiple nodes when more than one node in the same g-direction have been removed. For example, suppose in Figure 4.4 that all $\kappa > 1$ consecutive nodes between x and q in the g_1 and g_1^{-1} directions have been pruned out. Then, CTR establishes a new edge between x and q by computing $x * g_1^{\kappa+1} = q$, and $q * g_1^{-(\kappa+1)} = x$ in forward and backward directions, respectively.

There are also situations where our graph pruning algorithm results in multiple edges and self loops. The former occurs when a node to be pruned is part of a cycle of three nodes as shown in Figure 4.5(a). In this case, pruning node v and applying CTR will result in multiple edges between nodes u and w as illustrated in Figure 4.5(b). Furthermore, by pruning the node w in Figure 4.5(b), CTR produces a self-loop for the node u. Thus, if the CTR algorithm produces $|\mathbb{G}|/2$ self-loops for a node, then the node will be isolated from a connected component, and the resulting pruned BCG will be disconnected. Figure 4.6 summarizes the CTR algorithm for a 4-regular BCG with four generators g_1, g_2, and their inverses $g_3 = g_1^{-1}$ and $g_4 = g_2^{-1}$.

The advantages of the CTR algorithm can be summarized by (a) deterministic edge rewiring (new neighbors replacing pruned node are algebraically determined by group multiplication) (b) simplicity in computation (2×2 matrix multiplications), and (c) the consistency with the original BCGs connection rule. Because CTR uses the same connection rule as in BCGs, we expect structural similarities between the original and the pruned BCGs. We investigate these issues in the rest of the chapter.

```
Require: $x$ := pruned node
Require: $\mathcal{N}_x = \{v \mid d(u,v) = 1\}$          ▷ a set of neighbors of node $x$
Require: $V'$ := a set of nodes in a pruned BCG
 1: procedure CutThroughRewiring($x, \mathcal{N}_x, V'$)
 2:     for each $v \in \mathcal{N}_x$ do
 3:         $g \leftarrow v^{-1} * x$
 4:         while $v_{dest} \notin V'$ do
 5:             $v_{dest} \leftarrow v_{dest} * g$
 6:         end while
 7:         if $v_{dest} \neq v$ then
 8:             establish a directed edge from $v$ to $v_{dest}$
 9:         end if
10:     end for
11: end procedure
```

Figure 4.6: Algorithm for Cut-Through Rewiring.

4.3 BCG Pruning: Connectivity

In this section, we investigate the connectivity of pruned BCGs. Because our BCG Pruning algorithm does not involve any complex mechanism to prevent the occurrence of isolated nodes, it is important to see wether or not the proposed algorithm guarantees the connectivity of pruned BCGs. Obviously, as the number of pruned nodes increases, the graph disconnection probability of pruned graph increases. Nonetheless, analytical and simulation results showed that pruned BCGs remain connected even after pruning 80% to 90% of nodes from the original graph.

4.3.1 Analysis

We begin our graph connectivity analysis by considering two cases where the pruned BCGs are disconnected:

1. **Case 1 - Node isolation:** The case that a pruned BCG contains at least one node which has no neighbor (isolated). The *nodal isolation probability* is the probability that one or more nodes in a pruned BCG are isolated.

2. **Case 2 - Multiple connected components:** The case that a pruned BCG contains two or more connected components (disjoint connected subgraphs) of orders larger than 1 but less than $n_t - 1$ after pruning (n_t

Table 4.2: Notation.

Notation	Description
$\mathbf{C}$	Borel Cayley graph (BCG)
$\mathbf{C}^{\mathrm{P}}$	a pruned BCG
$\mathbf{C}^{\mathrm{E}}$	a expanded BGG
V	vertex set (or subgroups)
$V(G)$	vertex set of a graph G
$\mathbb{G}$	generating set of a BCG
g, g^{-1}	generator and its inverse
I	identity matrix
$\mathcal{N}_v$	set of a node v's immediate neighbors
n_o	size of the original BCG
n_t	target size of a pruned BCG
n	size of the benchmark graphs
m	number of nodes contained in g-direction
γ	number of pruned nodes
δ	number of isolated nodes
$P^i_\gamma(v,g)$	probability of a node v to have exactly i nodes in g-direction after pruning γ nodes and rewiring
$P^0_\gamma(v)$	probability of a node v to be an isolated node after pruning γ nodes and rewiring
$P(iso)$	nodal isolation probability
$P(dis)$	graph disconnection probability
$\Delta(iso)$	node isolation ratio
$\Delta(dis)$	graph disconnection ratio
$\mathbb{S}$	set of pruned BCG samples
$\mathbb{S}_{dis}$	set of disconnected pruned BCG samples
$\mathrm{diam}(G)$	diameter of a graph G
$\mu(G)$	average path length of a graph (G)
λ_2	algebraic connectivity

is the target size). In this case, we only consider the order of connected components greater than or equal to 2 since node isolation (component with order exactly 1) is considered in Case 1. The *multiple connected components probability* is the probability that there exists two or more connected components in the pruned BCG, excluding Case 1.

Based on the theoretical results presented in [61, 62], we expect the multiple connected components probability (Case 2) to decrease very quickly with the size of a graph. So in this thesis, we only derive analytical expressions for the graph disconnection probability due to node isolations (Case 1). Also, in the following sections, we will show that the multiple connected components probability (Case 2) is significantly less than the nodal isolation probability.

Lemma 4.1. Let $\mathbf{C}$ be a 4-regular Borel Cayley graph, and let m be the smallest integer such that $g^m \ (\mathbf{mod}\ p) = I$, where g is a generator, and I is the identity. If $\mathbf{C}^{\mathrm{P}}$ is a pruned BCG of $\mathbf{C}$ after pruning γ nodes by random pruning and rewiring using CTR, then the probability of a node $v \in V(\mathbf{C}^{\mathrm{P}})$ having exactly i distinct nodes in the g-direction,

$$P_\gamma^i(v,g) = \frac{\binom{\alpha}{\alpha - i}\binom{n_o - \alpha}{\gamma - (\alpha - i)}}{\binom{n_o}{\gamma}} \ , \tag{4.1}$$

for $(\alpha - i) \leq \gamma \leq n_o$ and $0 \leq i \leq \alpha$, where $\alpha = m - 1$, and n_o denotes the size of the original Borel Cayley graph.

Proof. Let $\Pi(v,g)$ be a set of distinct nodes in the g-direction excluding the starting node v, that is, $\Pi(v,g) = \{\pi \in V(\mathbf{C}^{\mathrm{P}}) \backslash \{v\} \mid \pi = v{*}g^q,\ \forall\, 1 \leq q \leq \alpha\}$. The random pruning algorithm selects γ nodes uniformly at random from the original Borel Cayley graph. Out of γ pruned nodes, if exactly $(\alpha - i)$ nodes are chosen (without replacement) from the set $\Pi(v,g)$, then there are exactly i nodes remaining in the g-direction of node v. Thus, the probability of a node $v \in V(\mathbf{C}^{\mathrm{P}})$ having exactly i distinct nodes in the g-direction after pruning γ nodes is given by the hypergeometric probability mass function (pmf) [63]

$$P_\gamma^i(v,g) = \Pr\left[\left|\Pi(v,g)\right| = i\right] = \frac{\binom{\alpha}{\alpha - i}\binom{n_o - \alpha}{\gamma - (\alpha - i)}}{\binom{n_o}{\gamma}} \ , \tag{4.2}$$

for $(\alpha - i) \leq \gamma \leq n_o$ and $0 \leq i \leq \alpha$. $\square$

Using Eq. (4.1), the probability of a node in a pruned BCG having exactly i nodes in the g-direction, we derive the nodal isolation probability after random pruning.

Lemma 4.2. Let $\mathbf{C}$ be a 4-regular Borel Cayley graph with generating set $\mathbb{G} = \{g_1, g_2, g_1^{-1}, g_2^{-1}\}$, and let m_1 and m_2 be the smallest integers such that $g_1^{m_1}(\mathbf{mod}\ p) = I$, and $g_2^{m_2}(\mathbf{mod}\ p) = I$. If we choose the generators satisfying $m_1 = m_2$, then the probability of a node $v \in V(\mathbf{C}^{\mathrm{P}})$ being an isolated node after pruning γ nodes and performing CTR

$$P(iso) = \frac{\gamma!}{n_o!}\,\frac{(n_o - 2\alpha)!}{(\gamma - 2\alpha)!} \ , \tag{4.3}$$

for $(2\alpha - i) \leq \gamma \leq n_o$ and $0 \leq i \leq 2\alpha$, where $\alpha = (m_1 - 1) = (m_2 - 1)$.

Proof. A node in a pruned BCG $v \in V(\mathbf{C}^{\mathrm{P}})$ is isolated if and only if it has no neighbor in the g_1 and g_2 directions. The reverse direction paths given by the inverse generators g_1^{-1} and g_2^{-1} are not considered because they only consist of the same nodes in the g_1 and g_2 direction paths, respectively. As in Lemma 4.1, the probability of a node $v \in V(\mathbf{C}^{\mathrm{P}})$ to be isolated is given by the hypergeometric pmf. By setting the number of the remaining nodes in both g_1 and g_2 directions to 0 (i.e., $i = 0$) and denoting $\alpha = m_1 - 1 = m_2 - 1$, the nodal isolation probability is given by

$$P(iso) = P_\gamma^0(v) = \frac{\binom{2\alpha}{2\alpha - 0}\binom{n_o - 2\alpha}{\gamma - 2\alpha - 0}}{\binom{n_o}{\gamma}} = \frac{\binom{n_o - 2\alpha}{\gamma - 2\alpha}}{\binom{n_o}{\gamma}} \tag{4.4}$$

$$= \frac{\gamma!}{n_o!}\,\frac{(n_o - 2\alpha)!}{(\gamma - 2\alpha)!}\ , \tag{4.5}$$

for $(2\alpha - i) \leq \gamma \leq n_o$ and $0 \leq i \leq 2\alpha$. □

Next, we derive the graph disconnection probability due to the node isolations (Case 1).

Lemma 4.3. Let δ be the number of isolated nodes in a pruned BCG, and let γ be the number of pruned nodes, then the graph disconnection probability of the pruned BCG due to the existence of one or more isolated nodes

$$P(dis) = 1 - \left[1 - \frac{\gamma!}{n_o!}\frac{(n_o - 2\alpha)!}{(\gamma - 2\alpha)!}\right]^{n_t}\ , \tag{4.6}$$

for $2\alpha \leq \gamma \leq n_o$, where $n_t = n_o - \gamma$ denotes the size of the pruned BCG.

Proof. We consider the probability of pruned BCG having $\delta = k$ isolated nodes to compute the graph disconnection probability. Since a graph containing at least one isolated node is disconnected, $1 - P_\gamma^0(v)$ is the probability of a node being connected (not isolated) in the pruned BCG after γ nodes pruning and rewiring. First, the probability of a pruned BCG having exactly k isolated nodes

$$\Pr[\delta = k] = \binom{n_o - \gamma}{k}\ \left[P_\gamma^0(v)\right]^k\ \left[1 - P_\gamma^0(v)\right]^{n_o - \gamma - k}\ , \tag{4.7}$$

and the probability that there is no isolated node in a pruned BCG

$$\Pr\left[\delta = 0\right] = \left[1 - P_{\gamma}^{0}(v)\right]^{n_t} \quad , \tag{4.8}$$

where the target size $n_t = n_o - \gamma$. Hence, the disconnection probability of a pruned BCG after γ nodes pruning and rewiring is given by

$$\begin{aligned} P(dis) = \Pr\left[\delta > 0\right] &= 1 - \Pr\left[\delta = 0\right] \\ &= 1 - \left[1 - \frac{\gamma!}{n_o!}\frac{(n_o - 2\alpha)!}{(\gamma - 2\alpha)!}\right]^{n_t} \quad . \end{aligned} \tag{4.9}$$

□

4.3.2 Validation

To verify the accuracy of the nodal isolation and graph disconnection probabilities (Eq. (4.3) and Eq. (4.6)), we resized the original BCGs of sizes $n_o = 1081$ and 5253 using the proposed BCG Pruning algorithm. The number of distinct nodes in the g_1 and g_2 directions were $m_1 = m_2 = 23$ and $m_1 = m_2 = 51$ for $n_o = 1081$ and 5253, respectively. The amount of pruned nodes, γ, was set to a multiple of 20 for $n_o = 1081$ and a multiple of 105 for $n_o = 5253$ (approximately 2% of the original graph size). For each γ, we generated $|\mathbb{S}| = 1500$ pruned BCG samples using the BCG Pruning algorithm. For the pruned BCG samples, we computed the node isolation ratio and the graph disconnection ratio.

The node isolation ratio was calculated as $\Delta(iso) = \frac{1}{n_t}\delta(\mathbf{C}^{\mathrm{P}})$, where $\delta(\mathbf{C}^{\mathrm{P}})$ is the number of isolated nodes in the pruned BCG and n_t is the size of the pruned BCGs. While the graph disconnection ratio was calculated as $\Delta(dis) = |\mathbb{S}_{dis}|/|\mathbb{S}|$, where $\mathbb{S}_{dis} = \{\mathbf{C}^{\mathrm{P}} \mid \delta(\mathbf{C}^{\mathrm{P}}) > 0,\ \mathbf{C}^{\mathrm{P}} \in \mathbb{S}\}$ (i.e., a set of *disconnected* pruned BCGs due to the node isolation). Note that the node isolation ratio and the graph disconnection ratio are the simulated results of the nodal isolation probability in Eq. (4.3) and graph disconnection probability in Eq. (4.6), respectively. Lastly, we computed the average node isolation ratio as $\overline{\Delta}(iso) = \frac{1}{|\mathbb{S}|\, n_t}\sum_{i=1}^{|\mathbb{S}|}\delta(\mathbf{C}_i^{\mathrm{P}})$, where $\mathbf{C}_i^{\mathrm{P}}$ is an instance of a pruned BCG.

In Figure 4.7, we compare the nodal isolation probability (analytical results) and the node isolation ratio (simulation results) of pruned BCGs. The nodal isolation probability remains below 0.01 until the normalized pruning quantity reaches 0.9 to 0.95. That is, pruning 90% $\sim$ 95% of nodes from the original BCGs produced fewer than $0.01n_t$ isolated nodes in pruned BCGs. In the figure, we also show the *target nodal isolation probability* at

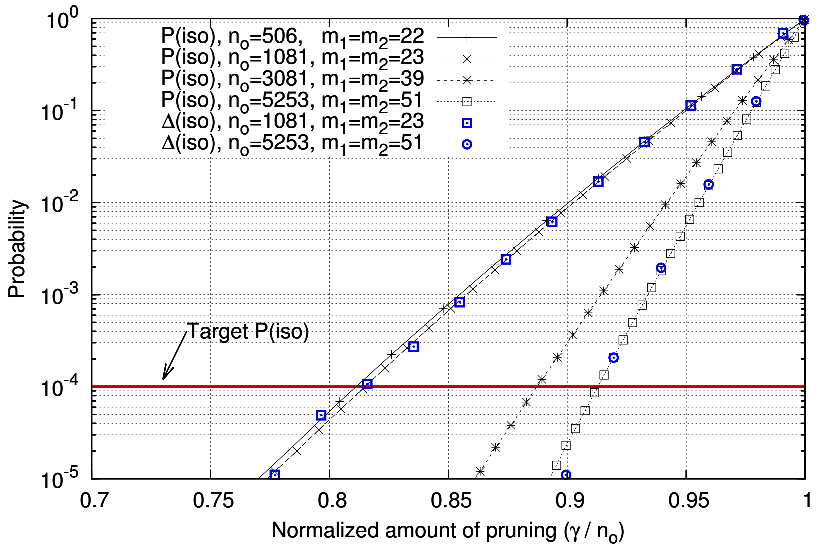


Figure 4.7: The nodal isolation probability $P(iso)$ and the average nodal isolation ratio $\overline{\Delta}(iso)$ of resized BCGs as a function of the normalized amount of pruned nodes γ/n_o. The average node isolation ratio was obtained by taking the average of the node isolation ratios of 1500 samples for each γ/n_o.

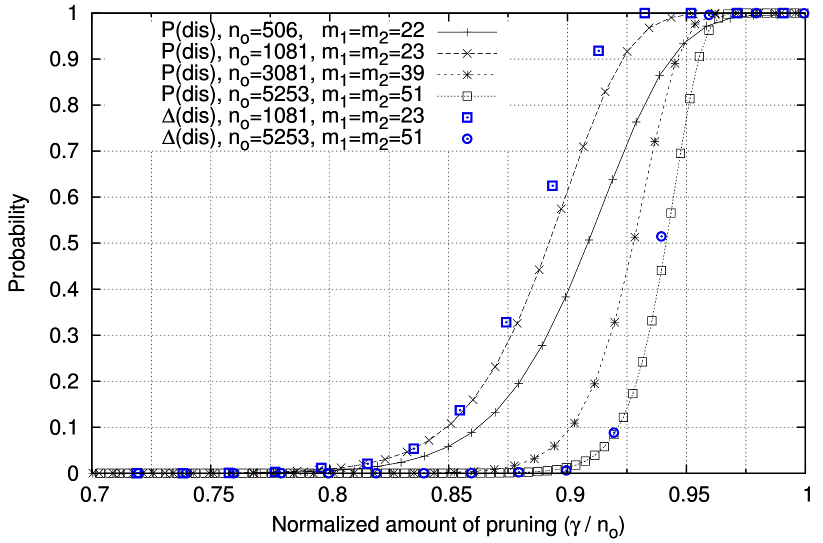


Figure 4.8: The graph disconnection probability $P(dis)$ and the ratio $\Delta(dis)$ as a function of the normalized amount of pruned nodes γ/n_o. The graph disconnection ratio was obtained by counting the number of disconnected graphs among 1500 samples for each γ/n_o.

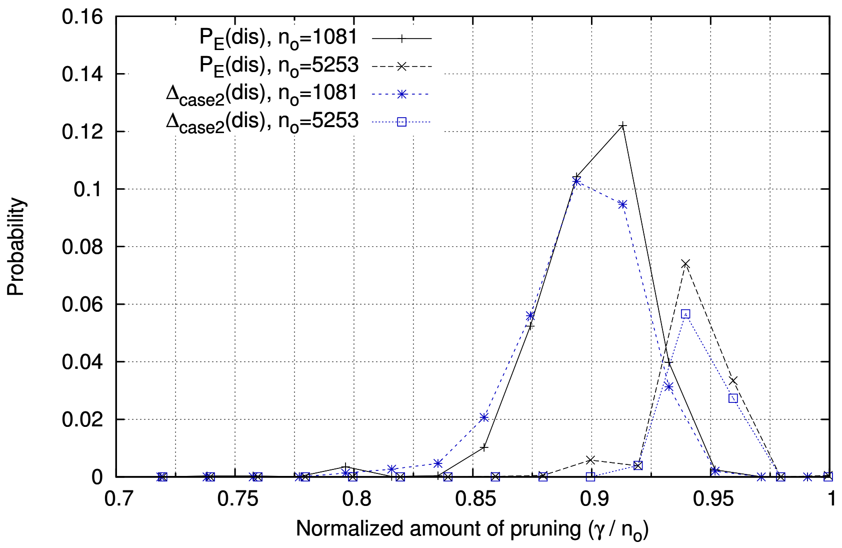


Figure 4.9: Analytical error $P_E(dis)$ and the Case 2 graph disconnection ratio $\Delta_{Case2}(dis)$ as a function of the normalized amount of pruned nodes γ/n_o.

$P(iso) = 0.0001$ [1] to illustrate how many node prunings the original BCG can tolerate before being disconnected with this target probability. The results show that up to 82% of nodes can be pruned from the $n_o = 506$ original BCG while maintaining the nodal isolation probability less than or equal to the target nodal isolation probability. The corresponding percentages of pruning that meet the target nodal isolation probability are 82%, 88%, and 92% for $n_o = 1081$, 3081, and 5253 BCGs, respectively.

Figure 4.8 shows the graph disconnection probability from Lemma 4.3 versus the graph disconnection ratio for BCGs of sizes $n_o = 1081$ and 5253. In the figure, we plotted the disconnection probability for $n_o = 506$ and 3081 as well. The analytical and simulation results show that the pruned BCGs are connected with very high probability for up to 80% pruning for $n_o = 1081$, and 90% for $n_o = 5253$. In fact, simulation results shows that 98.87% and 100% of pruned BCGs are connected when 80% of nodes are pruned from $n_o = 1081$ and $n_o = 5253$ BCGs, respectively. For 90% pruning, we observed from the simulation that 99.4% of the pruned BCGs with the original size $n_o = 5253$ are connected.

Even though our analytical expression for the graph disconnection probability $P(dis)$ in Eq. (4.6) only considers node isolations, we found the error

[1] Note that, this target nodal isolation probability $P(iso) = 0.0001$ guarantees at most one node isolation for $10,000$ nodes networks.

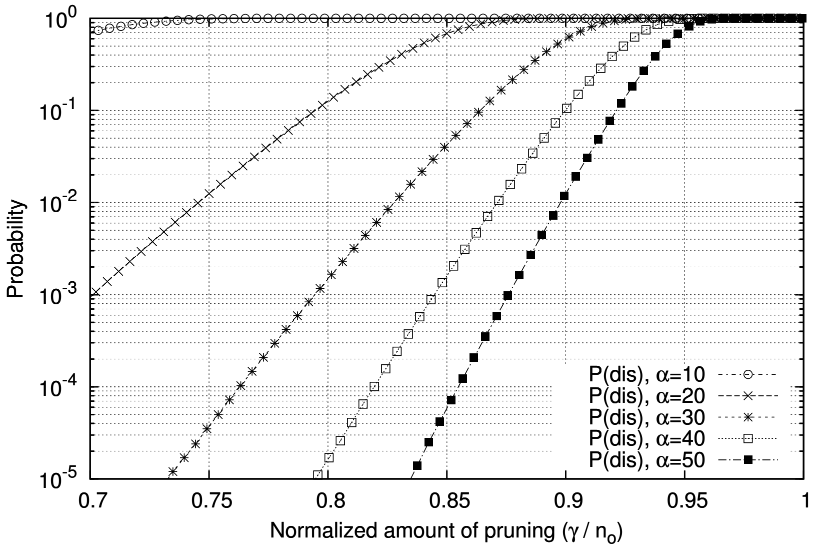


Figure 4.10: $P(dis)$ of the resized graphs as a function of γ/n_o for $\alpha = 10, \ldots, 50$. The size of the original BCG $n_o = 5253$.

between our analytical expressions and our simulation results to be small. Moreover, in Figure 4.9, we show the error $P_E(dis) = |P(dis) - \Delta(dis)|$ for pruned BCGs[2] of sizes $n_o = 1081$ and 5253. We found the role of the multiple connected components (Case 2), represented by $P_E(dis)$, become insignificant as n_o and n_t increase.

4.3.3 Relation to the graph generators

By definition of BCGs, the numbers of nodes contained along the g_1 and g_2 directions are given, respectively, by the integers m_1 and m_2 such that $g_1^{m_1}(\textbf{mod}\ p) = I$, and $g_2^{m_2}(\textbf{mod}\ p) = I$, where $0 \leq m_1, m_2 \leq k$, and I is the identity. Thus, m_1 and m_2 are determined by the choice of generators. Assuming that we choose two generators such that $m_1 = m_2$, the graph disconnection probability in Eq. (4.6) is a function of $\alpha = m_1 - 1 = m_2 - 1$ in addition to the number of pruned nodes. Moreover, α is determined by the generators because m_1 and m_2 are computed such that $g_1^{m_1}(\textbf{mod}\ p) = I$, and $g_2^{m_2}(\textbf{mod}\ p) = I$. Based on this observation, we investigate the relationship between α and the graph disconnection probability.

In Figure 4.10, we plotted $P(dis)$ as a function of γ/n_o for the pruned BCGs generated from the $n_o = 5253$ original BCGs. The maximum possible

[2] $P_E(dis)$ also represents Case 2 graph disconnection ratio.

value for α is 50 since the maximum values of m_1 and m_2 for this particular graph are limited by $k - 1 = 50$. We used different values for α to show effects of generators on the connectedness of the pruned BCGs. The results showed that $P(dis)$ of the pruned BCG is sensitive to α. In fact, the original BCGs formulated by generators with large α values experience fewer graph disconnections after pruning. Thus, it is desired to choose generators that yield large m_1 and m_2 to ensure lower graph disconnection probability after pruning.

4.4 BCG Pruning: Topological and Spectral Properties

In this section, we quantify the topological and spectral properties of pruned BCGs. More specifically, we compare the original BCGs and their pruned BCGs using the metrics including diameter, the average path length, and the algebraic connectivity defined in Chapter 2.

4.4.1 Graph generation

The original BCGs with $n_o = 1081$, 2211, 3081, 4063, and 5253 were generated and pruned by our BCG Pruning algorithm. Table 4.3 summarizes the sizes and parameters of the original and pruned BCGs.

We have $(p \times k - 1)$ generator candidates in total because any node element except the identity can be a generator. However, for generators $g_1 = \left(\begin{smallmatrix} a^{t_1} & y_1 \\ 0 & 1 \end{smallmatrix}\right)$, and $g_2 = \left(\begin{smallmatrix} a^{t_2} & y_2 \\ 0 & 1 \end{smallmatrix}\right)$, we simply set $y_1 = y_2 = 1$ since the parameters y_1 and y_2 do not affect the connections of BCGs. In addition, we set the range of t_1 and t_2 to $1 \leq t_1 < t_2 \leq 10$ instead of using all (t_1, t_2) pairs in the range $0 \leq t_1 < t_2 \leq (k - 1)$. Thus, the total number of generator combinations was 45, where each (t_1, t_2) pair generated a unique BCG with different connection pattern.

We arbitrarily set the target sizes $n_t = 1000$, 2000, 3000, 4000, and 5000. Note that n_t can be any arbitrary number in the range $(0, n_o]$. The amount of pruned nodes, $\gamma = n_o - n_t$.

4.4.2 Diameter

Since BCGs are known as one of the densest 4-regular graphs for a given range of diameters [64], we compare the diameters of the original and pruned BCGs. For a given BCG of size n_o, we computed the average diameter $(\overline{d})$

Table 4.3: Sizes and parameters of the original BCGs and corresponding target sizes.

n_o	p	k	a	n_t					samples
1081	47	23	2	1000					45
2211	67	33	6	1000	2000				45
3081	79	39	2	1000	2000	3000			45
4063	239	17	6	1000	2000	3000	4000		45
5253	103	51	2	1000	2000	3000	4000	5000	45

and standard deviation (σ) over n pruned BCGs of size n_t in Table 4.4. For example, $\overline{d} = 10.5$ for the $n_o = 5253$ BCGs became $\overline{d} = 9.3$ after pruning $\gamma = 4253$ nodes (i.e., $n_t = 1000$). More interestingly, we found that the average diameters of the pruned BCGs are relatively invariant with the amount of pruning. For instance, the average diameter of pruned BCGs with $n_t = 1000$ created from the original BCGs with $n_o = 5253$ is 9.3 which is close to $\overline{d} = 8.7$ of pruned BCGs with $n_t = 1000$ resized from the original BCGs with $n_o =$ 1081.

4.4.3 Average path length

In general, a small average diameter does not automatically guarantee a small average path length. Thus, we also evaluated the average path length of the original and pruned BCGs in Table 4.5. As expected, the average path lengths of the original BCGs were short (e.g., $\overline{\mu} = 5.6 \sim 7.4$). We found that, for a given target size n_t, the average path lengths of the pruned BCGs remain close to those of the original BCGs. For example, $6.2 \leq \overline{\mu} \leq 6.4$ for $n_t = 2000$ regardless of n_o.

4.4.4 Algebraic connectivity

The average algebraic connectivity $\overline{\lambda_2}$ of the original and pruned BCGs are summarized in Table 4.6. In the table, the larger the $\overline{\lambda_2}$ is, the better connected the graph is. As indicated, $\overline{\lambda_2}$ of the pruned BCGs remained close to that of the original BCGs. For instance, in most cases, the average algebraic connectivity values of the pruned BCGs were close to those of the original BCGs.

However, the average algebraic connectivity of the $n_o = 4063$ BCGs pruned to $n_t = 1000$ were decreased by half which implies the loose connectedness of the pruned BCGs. We believe this undesirable result is attributed to the relatively small parameter k ($k = 17$ in this case).

Table 4.4: Average diameter ($\overline{d}$) of the original and resized BCGs. The σ and n are the standard deviation and sample size, respectively.

n_o	$n_t = n_o$ $\overline{d}\ (\sigma, n)$	$n_t = 5000$ $\overline{d}\ (\sigma, n)$	$n_t = 4000$ $\overline{d}\ (\sigma, n)$	$n_t = 3000$ $\overline{d}\ (\sigma, n)$	$n_t = 2000$ $\overline{d}\ (\sigma, n)$	$n_t = 1000$ $\overline{d}\ (\sigma, n)$
1081	8.1 $(0.3, 45)$					8.1 $(0.3, 45)$
2211	8.9 $(0.6, 42)$				9.1 $(0.4, 42)$	8.7 $(0.5, 42)$
3081	9.4 $(0.9, 42)$			9.7 $(0.9, 42)$	9.1 $(0.3, 42)$	8.7 $(0.6, 42)$
4063	9.2 $(0.4, 45)$		10.0 $(0.2, 45)$	9.9 $(0.3, 45)$	9.3 $(0.5, 45)$	11.1 $(0.5, 39)$
5253	10.5 $(1.3, 42)$	10.6 $(1.2, 42)$	10.2 $(0.6, 42)$	10.0 $(0.3, 42)$	9.2 $(0.4, 42)$	9.3 $(0.9, 42)$

Table 4.5: Average of average path lengths ($\overline{\mu}$) of the original and resized BCGs. The σ and n are the standard deviation and sample size, respectively.

n_o	$n_t = n_o$ $\overline{\mu}\ (\sigma, n)$	$n_t = 5000$ $\overline{\mu}\ (\sigma, n)$	$n_t = 4000$ $\overline{\mu}\ (\sigma, n)$	$n_t = 3000$ $\overline{\mu}\ (\sigma, n)$	$n_t = 2000$ $\overline{\mu}\ (\sigma, n)$	$n_t = 1000$ $\overline{\mu}\ (\sigma, n)$
1081	5.6 $(0.1, 45)$					5.6 $(0.1, 45)$
2211	6.4 $(0.2, 42)$				6.3 $(0.1, 42)$	5.7 $(0.1, 42)$
3081	6.7 $(0.3, 42)$			6.7 $(0.2, 42)$	6.2 $(0.0, 42)$	5.7 $(0.1, 42)$
4063	6.9 $(0.1, 45)$		6.9 $(0.1, 45)$	6.6 $(0.1, 45)$	6.3 $(0.0, 45)$	6.3 $(0.0, 39)$
5253	7.4 $(0.5, 42)$	7.3 $(0.4, 42)$	6.9 $(0.2, 42)$	6.6 $(0.1, 42)$	6.3 $(0.0, 42)$	5.9 $(0.3, 42)$

Table 4.6: Average algebraic connectivity ($\overline{\lambda_2}$) of the original and resized BCGs. The σ and n are the standard deviation and sample size, respectively.

n_o	$n_t = n_o$ $\overline{\lambda_2}\ (\sigma, n)$	$n_t = 5000$ $\overline{\lambda_2}\ (\sigma, n)$	$n_t = 4000$ $\overline{\lambda_2}\ (\sigma, n)$	$n_t = 3000$ $\overline{\lambda_2}\ (\sigma, n)$	$n_t = 2000$ $\overline{\lambda_2}\ (\sigma, n)$	$n_t = 1000$ $\overline{\lambda_2}\ (\sigma, n)$
1081	0.7 $(0.2, 45)$					0.6 $(0.1, 45)$
2211	0.6 $(0.2, 42)$				0.5 $(0.1, 42)$	0.5 $(0.0, 42)$
3081	0.5 $(0.2, 42)$			0.5 $(0.2, 42)$	0.5 $(0.1, 42)$	0.5 $(0.1, 42)$
4063	0.6 $(0.1, 45)$		0.6 $(0.1, 45)$	0.5 $(0.0, 45)$	0.5 $(0.0, 45)$	0.3 $(0.0, 39)$
5253	0.4 $(0.2, 42)$	0.4 $(0.2, 42)$	0.5 $(0.1, 42)$	0.5 $(0.1, 42)$	0.5 $(0.0, 42)$	0.5 $(0.1, 42)$

Table 4.7: Sizes and parameters of the benchmark graphs.

toroidal & diagonal meshes				small-world networks		
n	l	k	samples	n	p_r	samples
1081	23	47	1	1000	0.2	52
2021	43	47	1	2000	0.2	43
3009	51	59	1	3000	0.2	33
4087	61	67	1	4000	0.2	19
5041	71	71	1	5000	0.2	31

Overall, our results indicate the diameter and average path length of the pruned BCG to be almost *pruning invariant* if 75% or less nodes are pruned from the original BCGs. These results imply that our BCG Pruning algorithm is effective in generating arbitrary size BCG while preserving the advantageous properties of the BCGs such as shorter diameter and average path length.

4.5 BCG Pruning: Information Dissemination Performance

This section investigates the information dissemination performance of pruned BCGs and benchmark graphs including the original BCGs, toroidal meshes, diagonal meshes, and small world networks.

4.5.1 Setups

The information dissemination performance of the pruned BCGs was compared against that of the original BCGs, toroidal and diagonal meshes [55], and small world networks (SWNs) over a wide range of network sizes from 1000 to 5000 nodes. The rewiring probability of the SWNs was fixed to $p_r = 0.2$ throughout the simulation.

The sizes of meshes were determined by $n = l \times k$, where l and k are odd numbers [55]. The SWNs were generated using the method described in [10] with a fixed rewiring probability, $p_r = 0.2$. Table 4.7 summarizes the parameters used in the toroidal and diagonal meshes, and SWNs. Note that because connections in the SWNs are random, we could obtain 19 to 52 connected SWNs out of 100 SWN realizations.

In all cases, we set the step size $\omega = (2d_{max}(\mathbf{C}^{\mathrm{P}}) + 1)$ to ensure asymptotic convergence of the system [53]. The initial states for all nodes were integers randomly chosen between -5 and 5 inclusive. We declared a graph to have reached an agreement once each node value equals the average of all initial values within a precision of 0.001. The performance of the consensus protocol was measured by the number of steps needed to reach a network-wide consensus. Obviously, a small number of steps implies a faster convergence rate. By comparing the number of steps to achieve a consensus for each of different network topologies, we gain valuable insights on the information dissemination performance of a given topology.

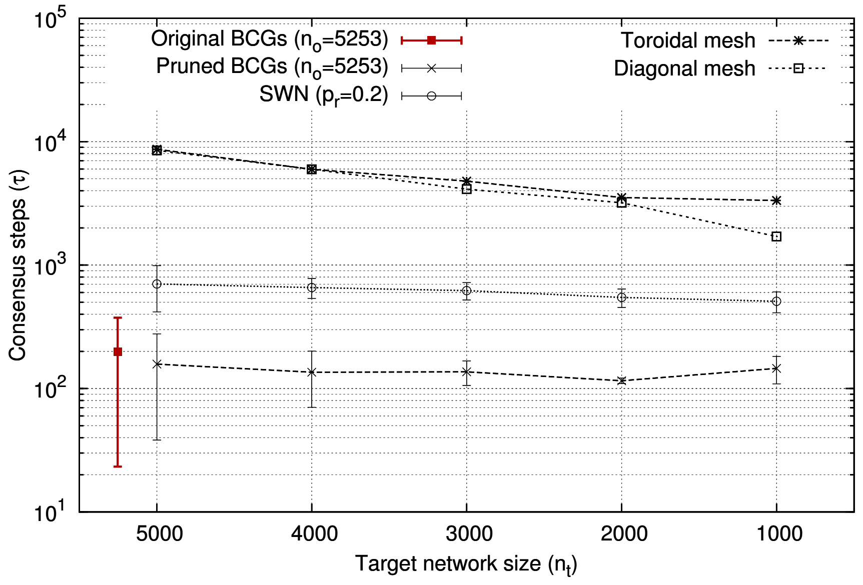
4.5.2 Evaluation

In Section 3.7, we saw that the regular ring lattices have the worst information dissemination performance among the regular graphs investigated. On the other hand, the (4-regular) toroidal and diagonal meshes performed better than the 4-regular ring lattices, while SWNs outperformed the meshes. Furthermore, we also observed that the original BCGs of comparable sizes performed better than the corresponding SWNs. Now, we compare the information dissemination performance of pruned BCGs with the original BCGs, toroidal meshes, diagonal meshes, and small world networks (SWNs).

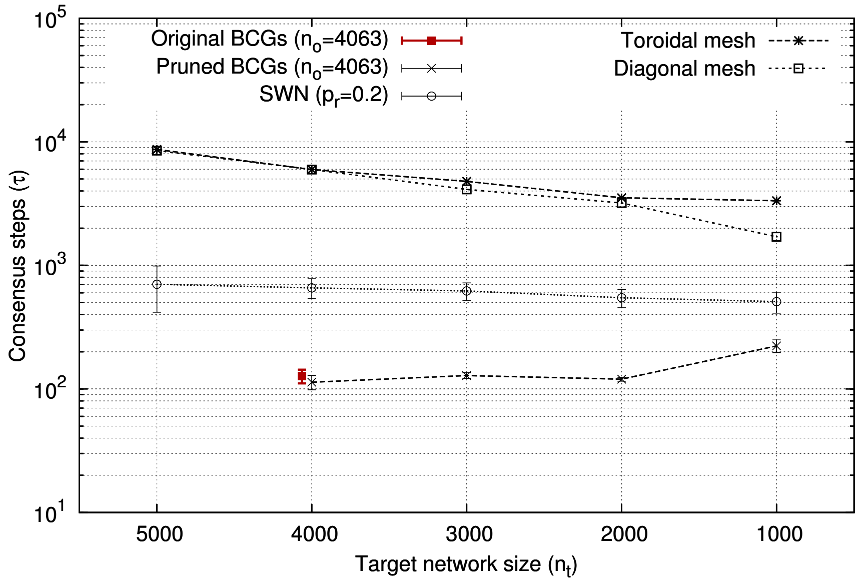
In Figure 4.11, we compare the information dissemination performance of the original and pruned BCGs, toroidal and diagonal meshes, and SWNs. The x-axis corresponds to a target size, n_t, ranging from 1000 to 5000, whereas the y-axis is the number of consensus steps required to reach the network-wide consensus. Each point in the plot is average consensus steps and corresponding standard deviation.

Figure 4.11(a) shows the information dissemination performance of the original and pruned BCGs when the original BCGs with $n_o = 5253$ are resized to the graphs with the target sizes between $n_t = 5000$ and 1000. Similarly, Figure 4.11(d) shows performance comparison between the original BCGs with $n_o = 2211$ and the pruned BCGs with $n_t = 2000$ and 1000. As indicated in the figures, the pruned BCGs performs almost identically to the original BCGs and with consistently superior performance over the toroidal and diagonal meshes, and SWNs.

Furthermore, from Figure 4.11, we observed that the information dissemination performance is relatively independent of the amount of pruning. For instance, the consensus steps of the pruned BCGs with $n_t = 1000$ span between 100 and 250, regardless of the size of the original BCGs. Even when a large number of nodes are pruned from the original BCG, the pruned BCGs of much smaller size still show better performance than that of the same size SWNs, or toroidal and diagonal mesh networks. For example, the pruned BCGs resized from $n_o = 5253$ to $n_t = 1000$ perform better than any other graphs as shown

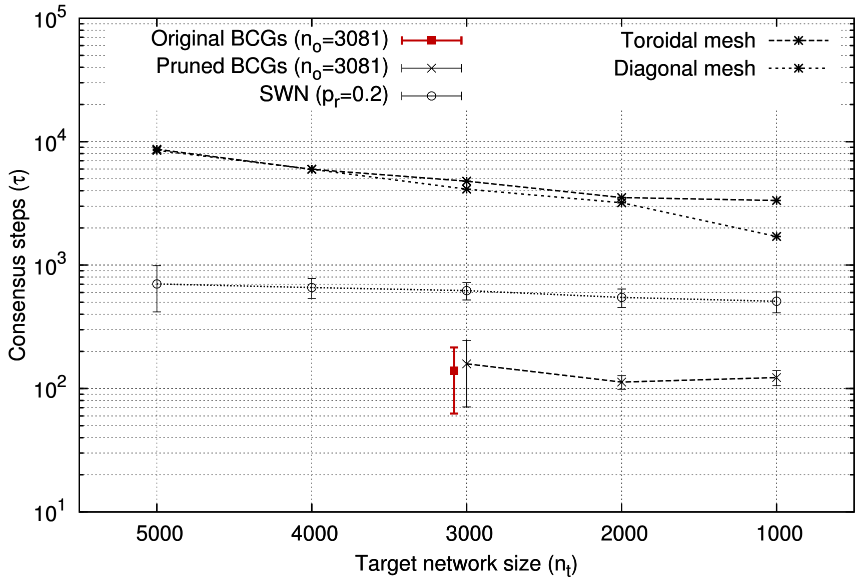


(a) $n_o = 5253$

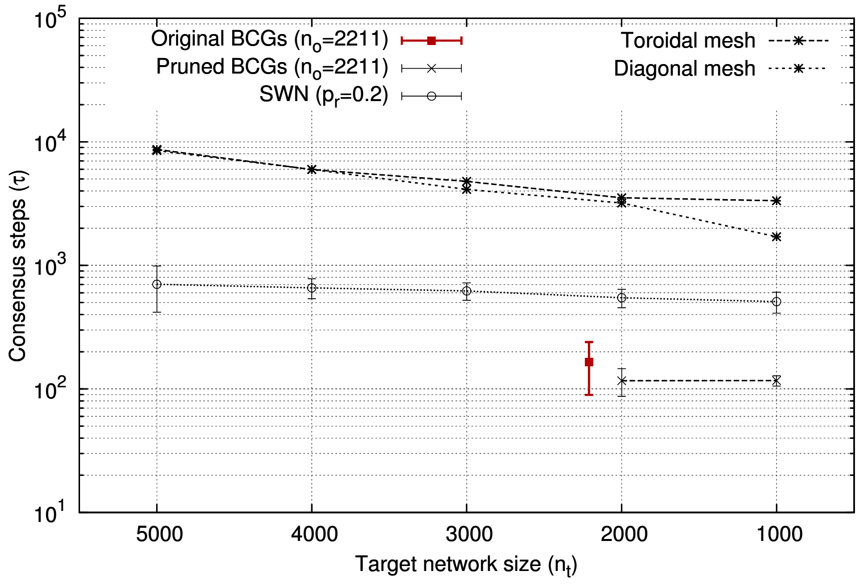


(b) $n_o = 4063$

Figure 4.11: Comparison of information distribution performance for the original and resized BCGs, toroidal and diagonal meshes, and SWNs as a function of network size. (continue)



(c) $n_o = 3081$



(d) $n_o = 2211$

Figure 4.11: Comparison of information distribution performance for the original and resized BCGs, toroidal and diagonal meshes, and SWNs as a function of network size.

in Figure 4.11(a). This confirms that the proposed BCG Pruning algorithm generated pruned BCGs that preserve the ultrafast information dissemination property of the original BCGs. Furthermore, such superior performance of pruned BCGs scales very well over the network sizes ranging from 1000 to 5000 nodes.

On the other hand, the information dissemination performance of the pruned BCGs with 1000 nodes that are resized from the original BCGs with 4063 nodes dropped by 74% in Figure 4.11(b). Such performance degradation can be explained by the relatively small $k(= 17)$ of the 4063 nodes original BCGs (see Table 4.3) together with our observations in Section 4.3.3. More specifically, a small k (and α that is upper bounded by $k - 1$) results in poor graph connectivity after pruning. The relatively small algebraic connectivity $\lambda_2 = 0.3$ of this particular graph in Table 4.6 supports our observation that a small k generally implies poor information dissemination. However, it is important to notice that even the worst performance observed in the simulations is still far better than that of the same size SWNs as shown in Figure 4.11.

4.6 BCG Random Expansion

This section presents a BCG expansion algorithm called the *BCG Random Expansion Algorithm*. Unlike the BCG Pruning algorithm, the proposed BCG Random Expansion algorithm expands the original BCG by adding nodes into the original BCG and by establishing edges between the injected nodes and pre-existing nodes. Simulation results showed that the Random Expansion algorithm expands the original BCGs with minimal penalty on the information dissemination performance, diameter, and average path length.

4.6.1 Terminologies

We refer to the BCG used as a base graph for expansion the *original BCG* or the *original graph*, interchangeably. Such original BCGs are denoted as $\mathbf{C}$. Similarly, we denote $\mathbf{C}^{\mathrm{E}}$ the *expanded BCG* or an *expanded graph* that is a graph expanded by our BCG random expansion algorithm. The size of the original and expanded BCGs are denoted by n_o and n_t, respectively. We denote the set of neighbors of a node $u \in V(G)$ as $\mathcal{N}_u$, where $V(G)$ and $E(G)$ are the node set and edge set of G, respectively. The *injected node* $\omega \in \Omega$ is a node that is added to the original BCG. In addition, Ω' defines the set of nodes to be injected in the original BCG.

Our random expansion algorithm operates as follows: First, two edges that involve four distinct nodes are selected. Second, the two edges are unwired

Table 4.8: The original BCGs and the target sizes.

parameters	values
original size (n_o)	253, 1081, 2211, 3081
target size (n_t)	1000, 2000, 3000, 4000, 5000

(disconnected). Finally, a new node is connected to these four distinct nodes. Details of three steps are described in the following sections.

4.6.2 Random expansion

Figure 4.12 illustrates how the random expansion algorithm injects a new node in the original BCG. To maintain the 4-regular property of the original BCG, the random expansion algorithm first randomly selects two distinct edges such that each edge joins a pair of nodes. Figure 4.12(a) shows an injected node ω and two randomly selected edges joining four nodes u, v, w, and x in the original BCG.

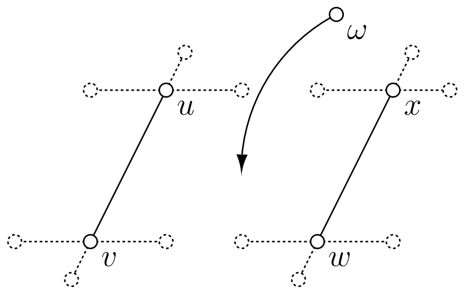
Once two edges have been selected, the random expansion algorithm disconnects those selected edges and establishes new links between the four nodes and the injected node. Figure 4.12(b) shows the injected node ω rewired with the original nodes u, v, w, and x.

Since we are not considering a multigraph depicted in Figure 4.12(c), we avoid multiple edges by selecting two edges involved with four *distinct* nodes. With this method, the BCG Random Expansion maintains the expanded BCGs to have the 4-regular property of the original BCGs.

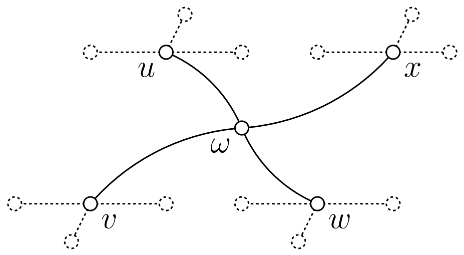
We summarize the proposed BCG Random Expansion algorithm in Figure 4.13. In the following sections, we evaluate the properties of expanded BCGs in terms of diameter, average path length, and information dissemination performance.

4.7 BCG Random Expansion: Topological Properties

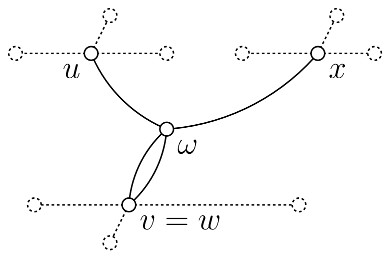
In this section, we compare the topological properties of expanded BCGs with those of the original BCGs, toroidal meshes, diagonal meshes, and Small World Networks (SWNs).



(a) BCG connections before injecting ω. Two edges $e(u, v)$ and $e(w, x)$ are randomly selected for node injection.



(b) BCG connections after injecting ω into the original BCG. New edges are established between ω and existing nodes u, v, w, and x.



(c) Formulation of multiple edge when the selected edges depicted in (a) are the same node, i.e., $v = w$.

Figure 4.12: Illustration of node injection and edge rewiring in BCG Random Expansion algorithm. Note that the original edges of nodes depicted in dotted line are omitted for brevity.

```
Require: $n_t > n_o$
Require: $\Omega' = \{w_1, w_2, \ldots, w_i\}, i = n_t - n_o$
 1: procedure BcgRandomExpansion($\mathbf{C}, \Omega'$)
 2:     $\mathbf{C}^{\mathrm{E}} \leftarrow \mathbf{C}$
 3:     $\Omega \leftarrow \emptyset$
 4:     for each $\omega \in \Omega'$ do
 5:         /* first edge selection and rewiring */
 6:         randomly choose $u \in V(\mathbf{C}^{\mathrm{E}})$                      ▷ root node
 7:         randomly choose $v \in \mathcal{N}_u$
 8:         $E(\mathbf{C}^{\mathrm{E}}) \leftarrow E(\mathbf{C}^{\mathrm{E}}) \setminus \{e_{u,v}\}$                ▷ unwire
 9:         $E(\mathbf{C}^{\mathrm{E}}) \leftarrow E(\mathbf{C}^{\mathrm{E}}) \cup \{e_{u,\omega}, e_{v,\omega}\}$          ▷ rewire
10:         /* second edge selection and rewiring */
11:         randomly choose $w \in V(\mathbf{C}) \setminus \{u, v\}$             ▷ root node
12:         randomly choose $x \in \mathcal{N}_w \setminus \{u, v\}$
13:         $E(\mathbf{C}^{\mathrm{E}}) \leftarrow E(\mathbf{C}^{\mathrm{E}}) \setminus \{e_{w,x}\}$                ▷ unwire
14:         $E(\mathbf{C}^{\mathrm{E}}) \leftarrow E(\mathbf{C}^{\mathrm{E}}) \cup \{e_{w,\omega}, e_{x,\omega}\}$          ▷ rewire
15:         $\Omega \leftarrow \Omega \cup \{\omega\}$
16:         $\mathbf{C}^{\mathrm{E}} \leftarrow \mathbf{C}^{\mathrm{E}} \cup \{\omega\}$
17:         if $|\Omega| (\mathbf{mod}\ n) = 0$ then
18:             $n \leftarrow |\mathbf{C}^{\mathrm{E}}|$
19:         end if
20:     end for
21: end procedure
```

Figure 4.13: Algorithm for BCG Random Expansion.

Table 4.9: Parameters of the benchmark BCGs.

n	p	k	a	t_1	t_2
253	23	11	2	3	7
1081	47	23	2	3	7
2211	67	33	6	5	7
3081	79	39	2	7	11
4063	239	17	6	3	7
5253	103	51	2	3	7

Table 4.10: Parameters of toroidal meshes, diagonal meshes, and SWNs.

toroidal and diagonal meshes				SWNs		
n	l	k	samples	n	p_r	samples
1081	23	47	1	1000	0.2	45
2021	43	47	1	2000	0.2	45
3009	51	59	1	3000	0.2	30
4087	61	67	1	4000	0.2	40
5041	71	71	1	5000	0.2	25

4.7.1 Setup

Using our BCG Random Expansion algorithm, the BCGs with the original sizes n_o = 253, 1081, 2211, and 3081 have been generated and expanded to target sizes n_t between 1000 and 5000. The BCG parameters p, k, and a have been chosen based on the guideline provided in Section 3.6. Table 4.8 summarizes the parameters of the original BCG. For each target size, 100 distinct expanded BCG samples (with variations on edge selections) have been obtained. For each expanded BCG sample, both the diameter and the average path length have been measured.

For performance comparison, we also evaluated the topological metrics of the benchmark BCGs, toroidal meshes, diagonal meshes, and SWNs. The benchmark BCGs summarized in Table 4.9 have been generated for comparison against the expanded BCGs. Also, Table 4.10 shows the sizes and parameters of benchmark graphs: toroidal meshes, diagonal meshes, and SWNs.

4.7.2 Diameter

In Figure 4.14, we plotted the average diameter $\overline{D}$ of the expanded BCGs versus the target network sizes. The diameter of the expanded BCGs is almost the same as that of the original BCGs regardless of the expansion factor $\gamma = n_t/n_o$. For example, the average diameter of the expanded BCGs with n_t = 5000 was $\overline{D} = 10$ for both $\gamma = 19.76$ ($n_o = 253$) and $\gamma = 4.63$ ($n_o = 1081$). Furthermore, regardless of γ, the expanded BCGs have the shortest diameters among the considered graphs other than the original BCGs.

4.7.3 Average path length

Figure 4.15 shows the average path length of the original and expanded BCGs together with the benchmark graphs. The results show that the average path length of the expanded BCGs are almost the same as those of the original BCGs over the whole range of target sizes. Moreover, the average path lengths of the expanded BCGs ($n_o = 253$) are shorter than those of toroidal meshes, diagonal meshes, or SWNs. In particular, regardless of γ, the expanded BCGs have at least 3.05, 2.70, and 1.02 times shorter average path lengths than those of toroidal meshes, diagonal meshes, and SWNs, respectively.

4.8 BCG Random Expansion: Information Dissemination Performance

In this section, we evaluate the information dissemination performance of the original and expanded BCGs, toroidal meshes, diagonal meshes, and SWNs.

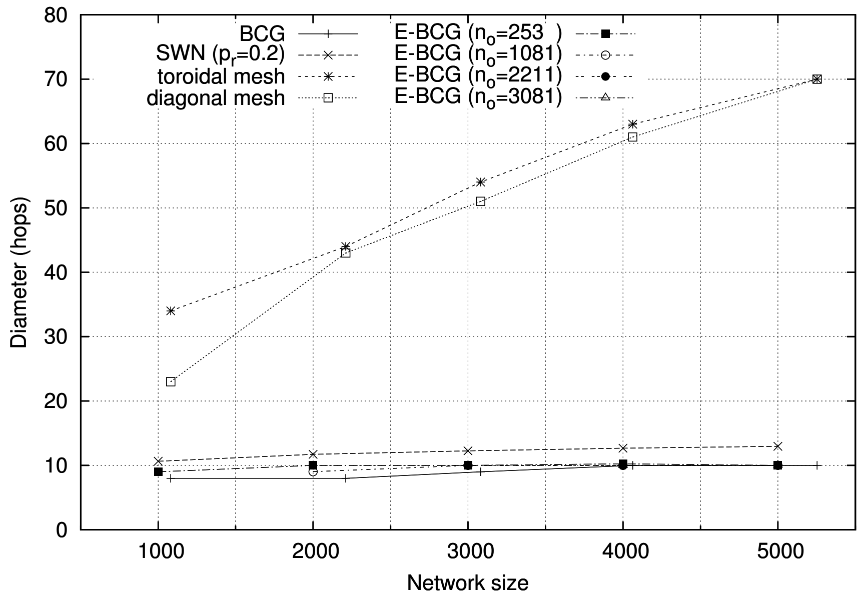
4.8.1 Setup and metrics

Information dissemination performance has been evaluated for the expanded BCGs, the original BCGs, toroidal meshes, diagonal meshes, and SWNs in terms of the number of required consensus steps τ. We declare that a consensus is reached if the state values of all nodes are identical within the precision of 10^{-3}. Obviously, the fewer the number of steps to reach a consensus, the better the information dissemination performance of a graph. We used the graph parameters summarized in Table 4.8, 4.9, and 4.10.

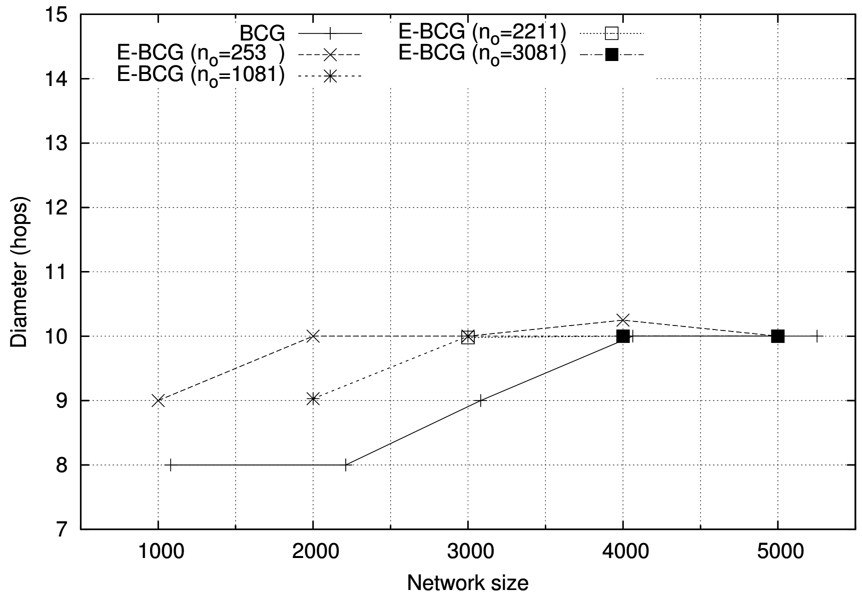
4.8.2 Performance

Figure 4.16 compares the information dissemination performance of the expanded BCGs ($n_o = 253, 1081$) against those of the BCGs, toroidal meshes, diagonal meshes, and SWNs. The consensus steps τ of the expanded BCGs are almost the same as those of the original BCGs with slight differences depending on the amount of expansion. Note that the standard deviation of the consensus steps for the expanded BCGs was negligible (e.g. less than four steps regardless of n_o).

Moreover, for $1.6 \leq \gamma \leq 19.8$, the expanded BCGs generated from the original BCG with $n_o = 253$ exhibit $24 \sim 72$, $15 \sim 70$, and $5 \sim 6$ times faster information dissemination performance than the toroidal meshes, diagonal meshes, and SWNs, respectively. Another interesting observation is that

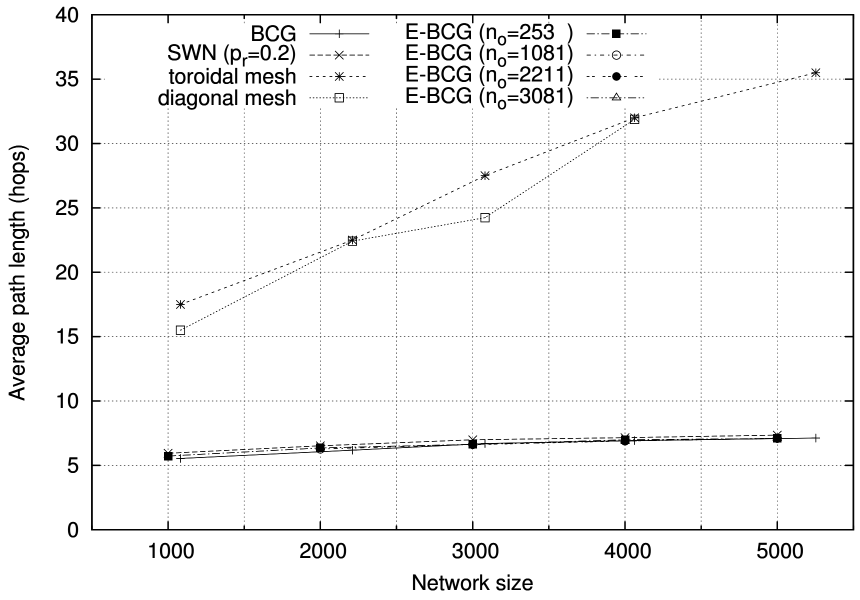


(a) Diameter of all graphs.

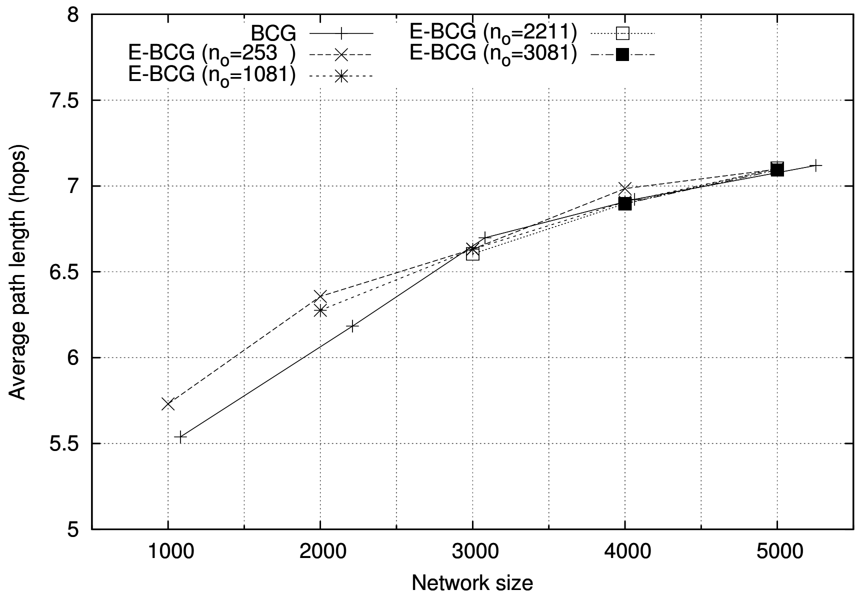


(b) Diameter of the original and expanded BCGs (zoomed the bottom part of Figure 4.14(a)).

Figure 4.14: Diameter of the expanded BCGs, the original BCGs, toroidal meshes, diagonal meshes, and SWNs.



(a) Average path length of all graphs.



(b) Average path length of the original and expanded BCGs (zoomed the bottom part of Figure 4.15(a)).

Figure 4.15: Average path length of the expanded BCGs, the original BCGs, toroidal meshes, diagonal meshes, and SWNs.

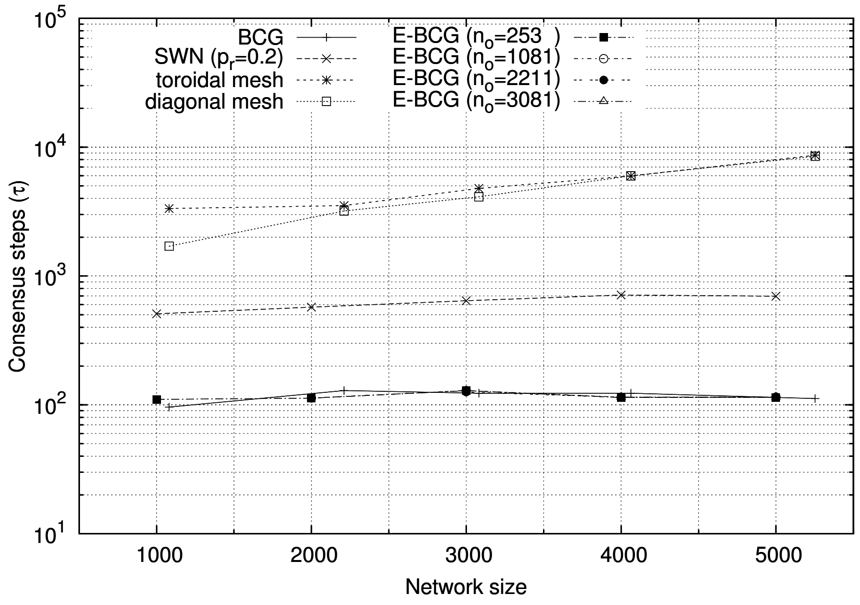


Figure 4.16: Information dissemination performance of the expanded BCGs, the original BCGs, toroidal meshes, diagonal meshes, and SWNs.

these expanded BCGs have slightly better information dissemination performance than the original BCGs for some γ's.

Overall, simulation results confirmed that the proposed BCG random expansion algorithm (i) maintains a small diameter, short average path length, and ultrafast information dissemination of the original BCGs in the expanded BCGs and (ii) is scalable over a wide range of expansion factors. In other words, our random expansion algorithm is effective in expanding the original BCG networks to larger networks of arbitrary sizes while maintaining the superior topological properties of the original BCGs.

Summary

We presented a graph theoretic approach that resolves the size limitation of BCGs. The proposed resizing method consists of two algorithms: the *BCG Pruning* and the *BCG Random Expansion* algorithms.

The BCG Pruning algorithm removes nodes from the original BCGs and rewires disconnected edges using the Cut-Through Rewiring (CTR) algorithm. Analytical analysis of the connectivity of pruned BCGs confirmed that the pruned BCGs are connected with high probability even after 80% $\sim$ 90% size reduction depending on the original BCG size. For example, when 80% of nodes are pruned from the original BCGs with $n_o = 1081$ and 5253, the pruned graphs are connected with the probabilities 0.9926 and 1.0, respectively. The accuracy of our analytical expressions were validated by extensive simulations as well.

Furthermore, we evaluated the topological and spectral graph properties of the original and pruned BCGs in terms of diameter, average path length, and algebraic connectivity. The results confirmed that the BCG Pruning algorithm guarantees strong structural similarity between the original and pruned BCGs. In fact, the proposed BCG Pruning algorithm preserves the small diameter, short average path length, and large algebraic connectivity of original BCGs. Most importantly, we evaluated the information dissemination performance of both the original and pruned BCGs for up to 5000 nodes. The results were promising: our proposed graph resizing algorithm generated pruned BCGs with comparable information dissemination performance of the original BCGs.

The BCG Random Expansion expands the original BCGs to arbitrary sizes. The simulation results showed that the proposed random expansion algorithm produces expanded BCGs with almost the same or superior information dissemination performance as well as favorable topological properties regardless of the expansion factor. In fact, even with an expansion factor $\gamma = 25$ (i.e., the original BCG is expanded 25 times larger), the diameter and the average path length were almost the same as those of the original BCG. For the same γ, the information dissemination performance of the expanded BCG was slightly better than that of the original BCG.

Chapter 5

Borel Cayley Graph Topology Control

5.1 Introduction

This chapter demonstrates an application of BCGs in wireless sensor networks (WSNs) by means of a topology control protocol. Because of the ad hoc nature of WSNs, a topology constructed without control is prone to be arbitrary rather than well constructed to provide optimal information distribution performance for WSN applications. Although WSNs demand different sets and levels of performance requirements depending on applications, the main goals of topology control protocol can be summarized as improving network lifetime, connectivity, information/data fusion speed, and routing efficiency.

As shown in previous chapters, the BCG has many advantageous properties when used as a communication graph. Those advantages include a small constant degree, a short average path length, and a small diameter over a wide range of network sizes. Most importantly, we showed that one can achieve fast information dissemination in a network when its underlying topology is a BCG or a Quasi BCG in Chapter 3. Comparative performance evaluation among BCGs, Small-World Networks, diagonal and toroidal meshes, and random graphs revealed that BCGs have the fastest information convergence over the range of network sizes considered.

However, using BCGs as topologies for wireless networks has been challenging because the BCG's random nature does not guarantee neighboring nodes to be within radio range. In fact, we must assume that all nodes in a network to be within a single hop communication range in order to use the BCGs for a wireless network topologies. This assumption limits the use of BCGs to very dense networks where all nodes are within one hop range of each other.

This chapter presents the *BCG Topology Control* (BCG-TC) for wireless sensor networks that builds network topology as close as possible to BCG topology while relaxing the one hop communication assumption. Specifically, we developed BCG-TC that constructs a communication topology for very dense wireless sensor networks of which the maximum nodal degree κ is constrained by a challenging value 4. We investigate the network connectivity and the topological properties of the communication graphs generated by BCG-TC, κ-NEIGH[1], and κ-Random (κ neighbors are randomly selected among the physical neighbors).

Furthermore, we evaluated the energy consumed by nodes to reach a global agreement using the average consensus protocol. Results showed that the topology generated by our proposed BCG-TC possesses superior performance in network connectivity when the transmission range is above a certain threshold. We also showed that nodes in BCG-TC networks consume the least amount of energy to reach a network-wide consensus among those in κ-NEIGH or κ-Random networks.

5.2 Borel Cayley Graph Topology Control

The one hop communication assumption in wireless networks is not practical since communication on BCG's long distance links would yield significant energy consumption and radio interference. In the following sections, we discuss how BCG-TC generates connected networks that resemble the original BCGs in topological properties while relaxing the one hop communication assumption. Also, the performance evaluation on information dissemination speed showed the BCG-TC networks to perform as well as the original BCGs.

5.2.1 Assumptions

We design BCG-TC such that the constructed network topologies are nearly identical to those of the original BCGs. Since we are using an undirected 4-regular BCG as a base communication graph, BCG-TC generates an undirected network topology with nodal degree less than or equal to four depending on the maximum radio range. The BCG-TC protocol is based on the following assumptions:

[1] Originally, the name of the protocol is k-NEIGH with k in English alphabet. However, we slightly abuse the name and use κ-NEIGH (with kappa) throughout the thesis to avoid confusion with the BCG parameter k.

- Maximum nodal degree $\kappa = 4$.
- High network density, for example, more than 1000 nodes in a 100m $\times$ 100m area.
- Stationary and identical nodes without sink nodes.
- Circular radio range with the maximum radius R. Each node can set its transmission power to reach neighbors within R.
- Nodes are assigned a unique ID between 0 to $n - 1$.
- Nodes are assigned the same BCG parameters p and k, and generator set $\mathbb{G} = \{g_1, g_2, g_1^{-1}, g_2^{-1}\}$.
- The order of BCG-TC operation is determined by node ID. No nodes are simultaneously performing BCG-TC operation.

Definition 5.1 (Physical neighbor)**.** A physical neighbor $\mathcal{N}_v^{phy}$ is a set of nodes such that each node in $\mathcal{N}_v^{phy}$ is within the communication range of node v.

Definition 5.2 (Logical neighbor)**.** A logical neighbor $\mathcal{N}_v^{log} \subset \mathcal{N}_v^{phy}$ is a set of physical neighbors of v such that there is a connection between v and each node in $\mathcal{N}_v^{log}$.

Definition 5.3 (BCG neighbor)**.** A BCG neighbor $\mathcal{N}_v^{bcg}$ is a set of nodes such that each node in $\mathcal{N}_v^{bcg}$ is an immediate neighbor of node v that is defined by BCG.

The BCG-TC protocol consists of two phases. During the first phase, all nodes in the network identify their BCG neighbors among physical neighbors. If one or more logical neighbors are found among the physical neighbors, the node establishes bidirectional links with those neighbors. After all nodes complete Phase-I, nodes having degree less than four start Phase-II to establish more connections. While the original BCG determines connections in the first phase, we use the Cut-Through Rewiring algorithm described in Section 4.2.1 during the second phase. In the followings, we describe the two phases of BCG-TC in detail.

5.2.2 BCG-TC Phase-I

Let x be a source node performing the BCG-TC Phase-I, then x starts Phase-I by broadcasting HELLO packet to its physical neighbors $\mathcal{N}_x^{phy}$. Each HELLO packet contains the source ID $srcid = x$ and a payload $pl = \mathcal{N}_x^{bcg}$. Each physical neighbor $w \in \mathcal{N}_x^{phy}$ receiving a HELLO packet extracts $\mathcal{N}_x^{bcg}$ from the packet and checks if its ID w matches one of the IDs in $\mathcal{N}_x^{bcg}$. If there is a match, then w establishes an edge with x and replies with an ACK packet containing source node ID $srcid = w$ and destination node ID $destid = x$. If it does not match, then w ignores the HELLO packet. After broadcasting HELLO packet, x resets the pre-defined timer T1 and collects ACKs until the timer T1 expires. From each received ACK, x extracts $srcid = w$ and checks if $w \in \mathcal{N}_x^{bcg}$. If there is a match, then x establishes an edge with w, otherwise, ignores the ACK. We summarize the Phase-I operation in Figure 5.1. Also, Figure 5.4 shows how physical neighbors process HELLO packets during Phase-I.

Although the topology of a network generated by Phase-I resembles that of the original BCG, depending on the maximum radio range R, it is possible that (i) some nodes are isolated (i.e., not connected to any nodes) and/or (ii) the whole network comprises multiple components (i.e., multiple disjoint subnetworks) of sizes larger than or equal to two[2]. In fact, when the maximum

```
Require: $\mathcal{N}_x^{bcg} = \{v \in V : v = x * g, \forall g \in \mathbb{G}\}$
Require: $\mathcal{N}_x^{log}$ := current logical neighbors of $x$
 1: procedure BcgTcPhaseI($x$,$\mathcal{N}_x^{bcg}$,$\mathcal{N}_x^{log}$)
 2:     if $|\mathcal{N}_x^{log}| < 4$ then                  ▷ $x$ performs Phase-I
 3:         $x$ broadcasts HELLO[$srcid = x$, $pl = \mathcal{N}_x^{bcg}$]
 4:         repeat
 5:             if $x$ receives ACK[$destid = x$, $srcid = w$] then
 6:                 if $w \in \mathcal{N}_x^{bcg}$ then
 7:                     $\mathcal{N}_x^{log} \leftarrow \mathcal{N}_x^{log} \cup \{srcid\}$
 8:                 end if
 9:             end if
10:         until T1 expires
11:     end if
12: end procedure
```

Figure 5.1: Algorithm for BCG-TC Phase-I.

[2] An isolated node, that is a node with no neighbor, is a network with size equal to one. Thus, a network with multiple components of which sizes are larger than or equal to two is disconnected network without isolated nodes.

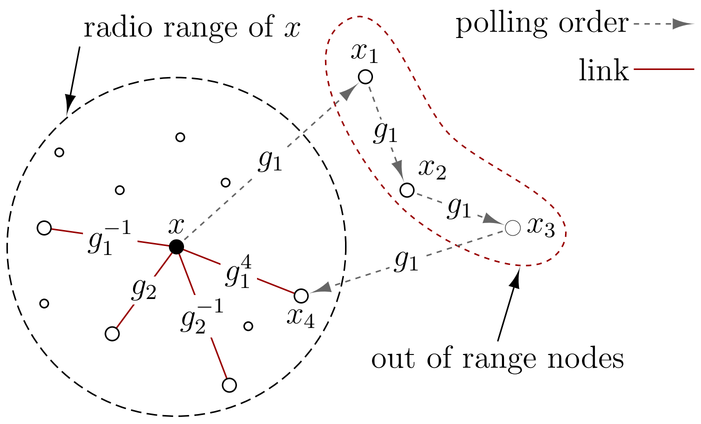


Figure 5.2: BCG-TC Phase-II example. After Phase-I, x failed to connect to the original neighbor x_1 in g_1 direction. By performing Phase-II, x connects to x_4.

radio range R is smaller than a certain threshold, the generated topology from Phase-I can be disconnected due to isolated nodes, multiple components, or a combination of both.

5.2.3 BCG-TC Phase-II

The second phase of BCG-TC enhances the network connectivity by systematically adding more edges to nodes having less than four neighbors after finishing Phase-I. From the connection rule of BCG, each node in a 4-regular BCG has four neighbors in g_1, g_2, g_1^{-1} and g_2^{-1} directions. During Phase-II, if a node is missing a neighbor in any g-direction, the node uses the CTR algorithm described in Section 4.2.1 to find the next available neighbor in the same g-direction.

Definition 5.4 (Empty g-direction)**.** An *empty g-direction* is a g-direction to where a node does not have any connected neighbor.

Let x be a node that found less than four neighbors during Phase-I. After all nodes complete Phase-I, x starts the following Phase-II operation to improve its connectivity. First, x identifies a set of empty g-directions $\mathbb{G}^e$. For each $g \in \mathbb{G}^e$, x computes the next available neighbor $w = x * g^2$, resets a pre-defined timer T2, and sends a HELLO packet to w. After sending the HELLO packet, x waits for ACK until T2 expires. If x receives an ACK from w, then it establishes a connection with w. Otherwise, x starts over and polls the next neighbors

```
Require: $\mathbb{G}^e$ := a set of empty $g$-directions
 1: procedure BcgTcPhaseII($x, y, g$)                         ▷ $x$ performs Phase-II
 2:     for each $g \in \mathbb{G}^e$ do
 3:         $i \leftarrow 2$
 4:         $w \leftarrow x * g^i$
 5:         while $w \neq x$ do
 6:             $x$ sends HELLO[$destid = w, srcid = x$]       ▷ $x$ polls $w$
 7:             repeat
 8:                 if $x$ receives ACK[$destid = x, srcid = w$] then
 9:                     $\mathcal{N}_x^{log} \leftarrow \mathcal{N}_x^{log} \cup \{w\}$          ▷ $w$ is a new neighbor
10:                     $x$ terminates Phase-II for $g$-direction
11:                 end if
12:             until T2 expires
13:             $i \leftarrow i + 1$
14:             $w \leftarrow x * g^i$
15:         end while
16:     end for
17: end procedure
```

Figure 5.3: Algorithm for BCG-TC Phase-II.

obtained by computing $x * g^3$, $x * g^4$, and so on, until it receives an ACK. We will discuss this polling order in the next section.

For example, let us consider the Phase-II operation of node x in Figure 5.2. Assuming that node x failed to find its BCG neighbor x_1 in its g_1-direction during Phase-I, x starts Phase-II operation by sending HELLO packet to its next available neighbor x_2 in the g_1-direction and waits for an ACK. Since x_2 and x_3 are out of range, x will not receive any ACK from them. When x finally receives ACK from $x_4 = x * g_1^4$ located within its radio range, it establishes a symmetric link with x_4 only when x_4 completed Phase-I but has less than four neighbors. Phase-II algorithm is summarized in Figure 5.3. The Phase-II operation is performed multiple times for each of missing neighbors that are not found during the Phase-I.

5.2.4 Neighbor polling order

Suppose BCG-TC uses 4-regular BCGs generated by $\mathbb{G} = \{g_1, g_2, g_1^{-1}, g_2^{-1}\}$. If a node x failed to find its neighbor in the g_1-direction, then, during the Phase-II, x polls its potential neighbors one by one until it finds one within its radio range. As described in Section 4.2.1, the CTR algorithm computes *ordered* set

```
Require: $\mathcal{N}_w^{log}$ := current logical neighbors of $w$
 1: procedure BcgTcProcHello($w$, HELLO)
 2:     if $|\mathcal{N}_w^{log}| = 4$ then                ▷ already has four neighbors
 3:         ignore HELLO
 4:     end if
 5:     if $w$ is performing Phase-I then
 6:         $w$ extracts $srcid = x$ and $pl = \mathcal{N}_x^{bcg}$
 7:         if $w \in \mathcal{N}_x^{bcg}$ then
 8:             $\mathcal{N}_w^{log} \leftarrow \mathcal{N}_w^{log} \cup \{x\}$
 9:             $w$ sends ACK[$destid = x$, $srcid = w$]
10:         else
11:             ignore HELLO
12:         end if
13:     else if $w$ is performing Phase-II then
14:         if $\exists\, g \in \mathbb{G}, h > 1$ s.t. $srcid = x * g^h$ then
15:             $\mathcal{N}_x^{log} \leftarrow \mathcal{N}_x^{log} \cup \{srcid\}$
16:             $w$ sends ACK[$y, x$]
17:         end if
18:     end if
19: end procedure
```

Figure 5.4: Algorithm for processing HELLO.

of potential neighbors using the BCG connection rule. For example, node x's polling order is computed by $x_1 = w * g_1$, $x_2 = w * g_1^2$, $x_3 = w * g_1^3$, and so on, as depicted in Figure 5.5. The polling order of x's g_1-direction is denoted by $\vec{x}(g_1) = (x_1 x_2 \ldots x_{\alpha-2} x_{\alpha-1})$. Note that, the inverse generator, g_1^{-1}, yields the same set of neighbors in reverse order $\vec{x}(g_1^{-1}) = (x_{\alpha-1} x_{\alpha-2} \ldots x_2 x_1)$.

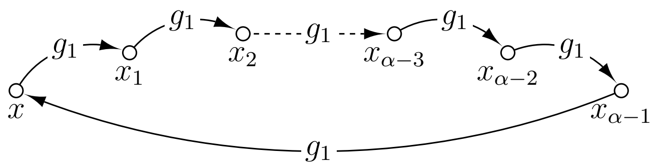


Figure 5.5: Illustration of $\alpha - 1$ nodes in x's g_1-direction polling order $\vec{x}(g_1)$.

Termination of polling order. Let α and β be positive integers such that $g_1^\alpha = I$ and $g_2^\beta = I$, respectively. Then, each node of a BCG has $\alpha - 1$ and $\beta - 1$ nodes in its cycles for g_1 (or g_1^{-1}), and g_2 (or g_2^{-1}) directions, respectively. Thus, the polling order in any g-direction terminates at the originating node

Table 5.1: Simulation parameters.

Parameters	Values
area dimension	100m $\times$ 100m
number of nodes (n)	1081
network density (ρ)	0.1081 nodes/m^2
max. radio range (R)	5, 10, 15, ..., 140 m
node distribution	uniformly randomly distributed in the area
sample size (n_s)	100 samples per radio range
BCG parameters	$p = 47$, $k = 23$, $a = 2$, $t_1 = 3$, $t_2 = 7$

(self-node). For example, the polling order of x's g_1-direction ends at the originating node (i.e., x) in Figure 5.5. The values of α and β are determined by the choice of generators as described in Section 4.3.3.

5.3 Performance Evaluation

In this section, we investigate the node isolation ratio, connectivity, diameter, average path length, and energy consumption of topologies generated by BCG-TC, κ-Neigh (Phase 1)[3], and κ-Random. The κ-Neigh networks (i.e., the networks constructed by κ-Neigh protocol) are generated for $\kappa = 4$ and 6. Since κ-Neigh with $\kappa = 4$ did not produce any connected networks over the whole radio ranges, we were not able to evaluate diameter, average path length, or energy consumption for those disconnected networks. Instead, we generated κ-Neigh networks with $\kappa = 6$ to evaluate these metrics. With the κ-Random protocol, connections between physical neighbors are established randomly with only one constraint, the maximum nodal degree $\kappa \leq 4$. For each of topology control protocols and radio range considered, we generated 100 network samples with random node deployments. Table 5.1 summarizes the simulation parameters.

5.3.1 Network connectivity

We investigate the network connectivity as a function of the maximum radio range R that is the maximum reachable distance between any two nodes. We declare a network to be connected if it has no isolated nodes and consists

[3] The κ-Neigh protocol has Phase 2, an optional stage to prune some excessive edges, but we did not use Phase 2 due to the same reason (complexity in implementation) as explained by the original authors.

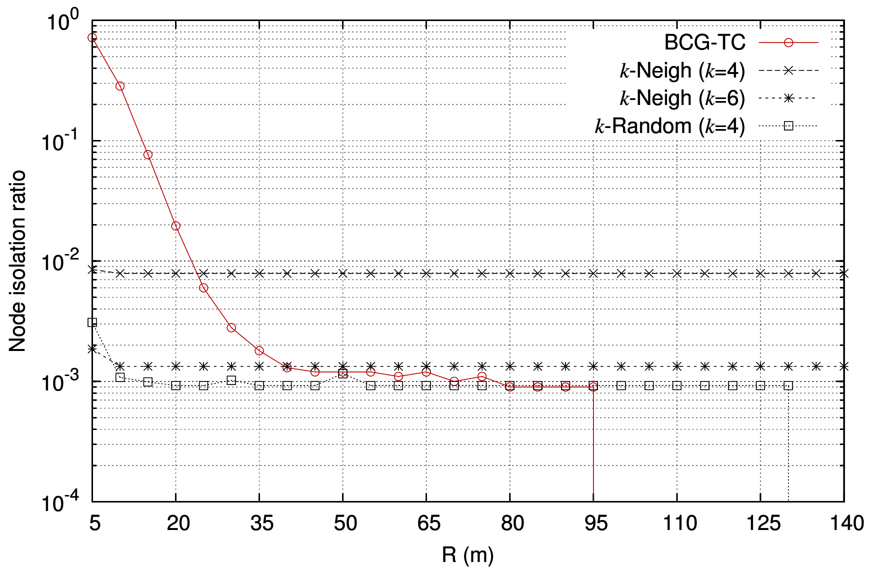


Figure 5.6: Average node isolation ratio versus the maximum radio range.

of one connected component (see Definition 2.7 and Definition 2.8). So, we evaluate network connectivity with the (i) *node isolation ratio*, the ratio of isolated nodes to the number of total nodes in a constructed network sample s_i,

$$\Delta(iso)_i = \frac{\text{number of isolated nodes in } s_i}{\text{number of nodes in } s_i} \ ,$$

and (ii) *connected network ratio*, the ratio of fully connected networks among all sample networks,

$$\Delta(con) = \frac{\text{number of connected networks}}{\text{total number of samples}} \ .$$

Node isolation. Figure 5.6 shows the average node isolation ratio $\overline{\Delta}(iso) = 1/n_s \sum_{i=1}^{n_s} \Delta(iso)_i$ versus the maximum radio range R. The κ-Neigh networks with $\kappa = 4$ shows a relatively large average node isolation ratio for all R. We found the average node isolation ratio of BCG-TC networks to be very large for $R < 20$m, but to decrease quickly to that of κ-Neigh with $\kappa = 4$ as R exceeds 25m. Moreover, $\overline{\Delta}(iso)$ of BCG-TC networks approaches that of κ-Neigh with $\kappa = 6$ as R exceeds 40m and finally becomes zero for $R \geq 100$m. On the other hand, $\overline{\Delta}(iso)$ of κ-Neigh networks with $\kappa = 6$ remains at 0.0013 for all $R > 5$m. Overall, BCG-TC networks display better node isolation ratio when $R \geq 40$m comparing to that of κ-Neigh with $\kappa = 4$ or 6.

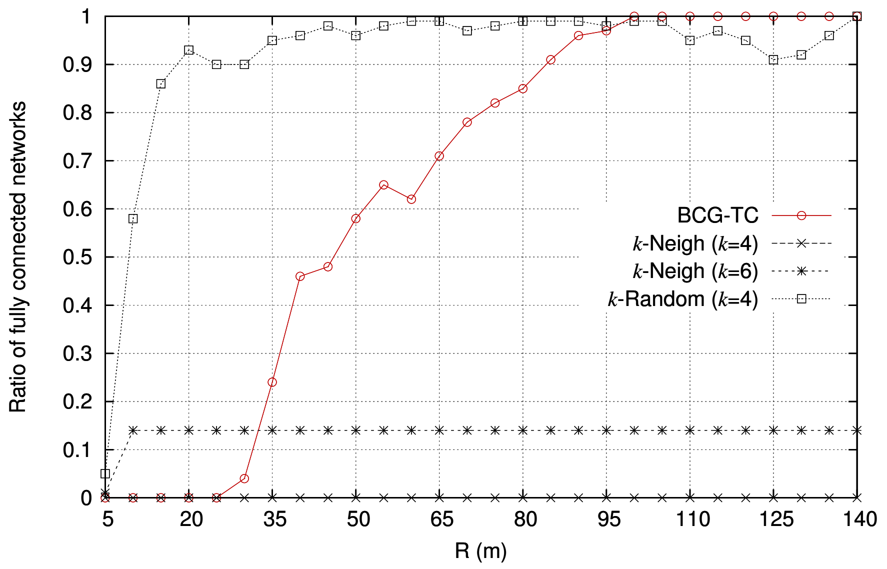


Figure 5.7: Connected network ratio versus the maximum radio range.

Network connectivity. Besides isolated nodes, a network can be disconnected due to multiple connected components. Although a network may not have any isolated nodes, multiple connected components can result in a disconnected network. Figure 5.7 shows the ratio of connected networks $\Delta(con)$ against R. Simulation results confirmed that κ-NEIGH with $\kappa = 4$ does not produce any connected network as shown in Figure 5.7, where all disconnected networks contain isolated node in Figure 5.8. Similarly, κ-NEIGH with $\kappa = 6$ generated disconnected networks with very high ratio 0.856, and 32% of disconnected networks did not contain any isolated nodes.

On the other hand, BCG-TC starts to construct connected networks from $R = 30$m, and $\Delta(con)$ consistently increases with R in Figure 5.7. As shown in Figure 5.8, all disconnected BCG-TC networks observed for $R \geq 40$m were disconnected due to several isolated nodes rather than multiple components. In fact, the average number of isolated nodes in BCG-TC networks was negligible for $R \geq 35$m as shown in Figure 5.8. For example, among 1081 nodes in BCG-TC networks, less than two nodes were isolated for $R \geq 35$m.

Moreover, Figure 5.9 shows that BCG-TC does not generate a disconnected network with multiple components for almost all R. In fact, only two networks among 100 network samples had multiple components. This confirms that node isolation is the main reason for a BCG-TC network to be disconnected. From these observations, we conclude that the *giant component*[4] of BCG-TC

[4] A giant component is a connected subgraph of G that consists of the majority of nodes

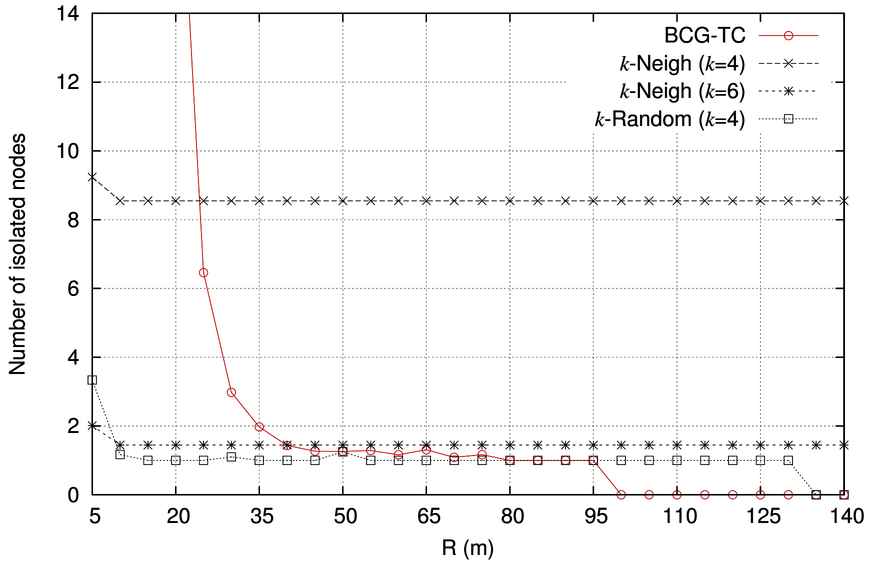


Figure 5.8: Average number of isolated nodes in a network.

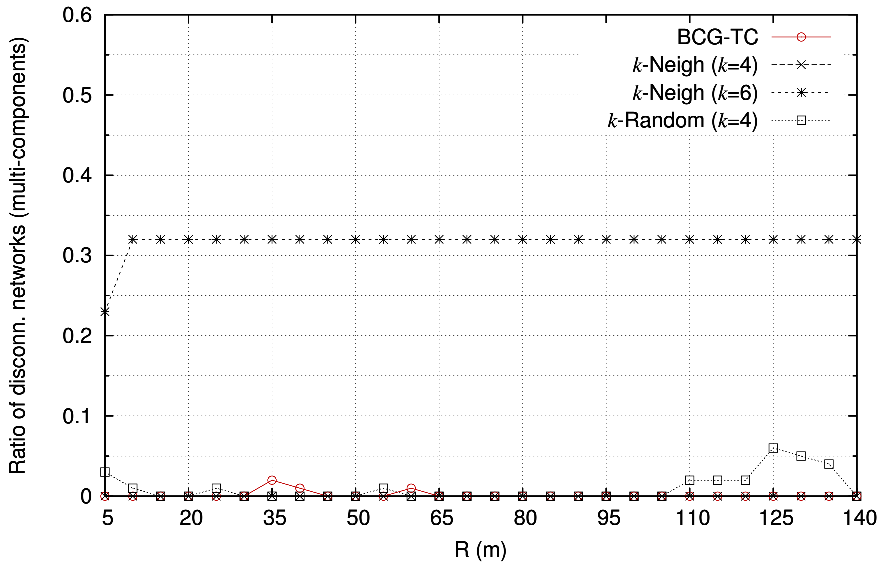


Figure 5.9: The ratio of disconnected networks without isolated nodes. Note that only the number of networks that are disconnected by multiple components but not by node isolations.

networks contain 99.8% of all nodes.

in a graph.

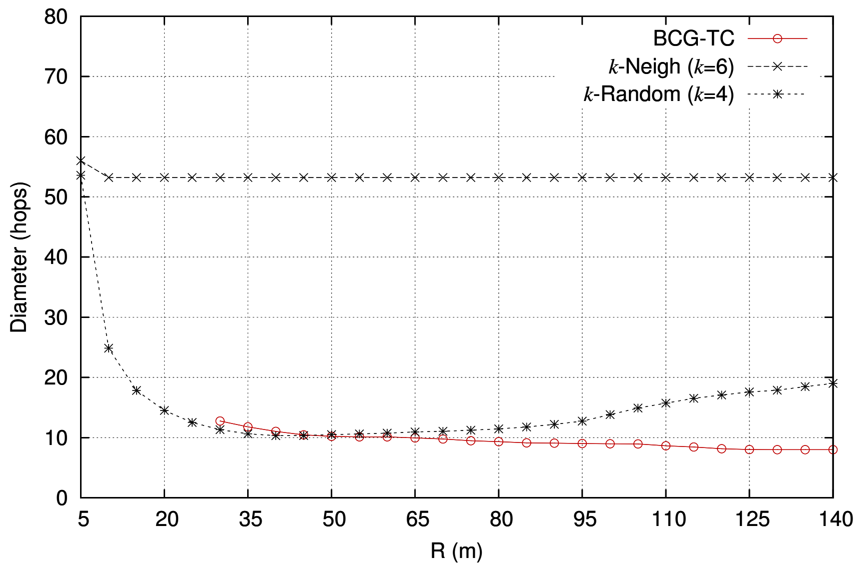


Figure 5.10: Radio range versus average diameter.

5.3.2 Topological properties

In this section, we take a closer look at the topological properties of the BCG-TC networks including diameter and average path length. Note that we only consider connected networks to evaluate diameter and average path length. Since we were not able to obtain connected network from κ-NEIGH with $\kappa = 4$, we only considered κ-NEIGH networks with $\kappa = 6$ for evaluating the diameter and average path length. Also, we did not consider BCG-TC results for $R < 30$ since BCG-TC did not produce any connected networks.

Diameter. In Figure 5.10, we plot the average diameter $\overline{D}$ for 100 network samples generated by BCG-TC, κ-NEIGH , and κ-random. For networks produced by κ-NEIGH with $\kappa = 6$, the average diameter remains around $\overline{D} = 53$ for whole $R > 5$m. While the average diameter of the networks generated by BCG-TC consistently decreases from 12.75 hops to 8 hops as R increases.

Average path length. The short average path length is another promising property of BCG-TC networks. We show $\overline{\mu}$, the sample mean of average path lengths, in Figure 5.11. Along with the short diameter illustrated in Figure 5.10, the BCG-TC networks display a very short average path length $\overline{\mu}$ throughout the maximum radio ranges considered.

In this section, we showed that, although nodal degree is constrained by a small value $\kappa = 4$, BCG-TC networks can achieve a small diameter and a short

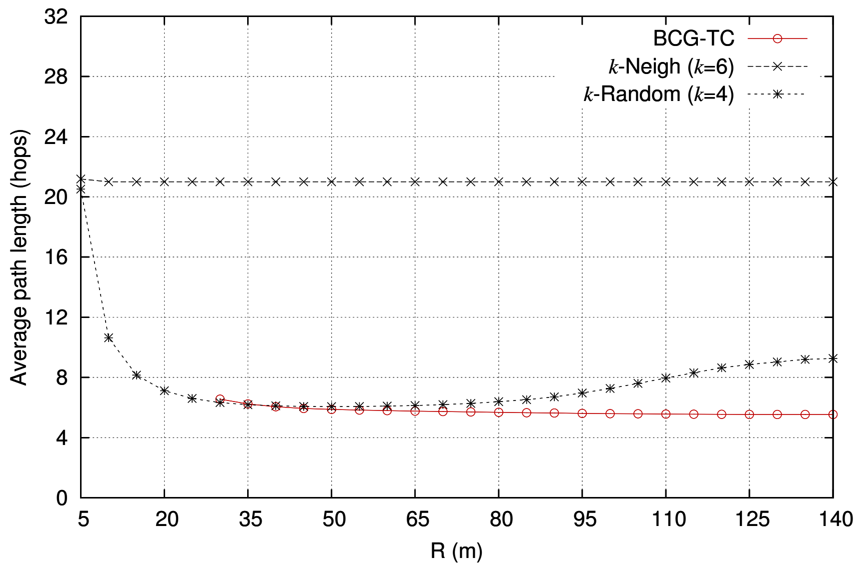


Figure 5.11: Radio range versus the average path length.

average path length when the maximum radio range $R > 35$m. Since not all WSN applications require fully connected networks, if we consider applications allowing a few isolation nodes, then BCG-TC networks can be considered as connected networks (without multiple components) for $R > 35$m. In the next section, we consider the energy efficiency of these topologies with the average consensus protocol.

5.4 Energy Consumption

In this section, we compute the energy needed for all nodes in a network to reach a consensus using the average consensus protocol. We use the simulation setup summarized in Table 5.1. In addition, we also applied the radio model described in [65]. For the sake of simplicity, we only consider the energy consumption from data transmission ($\mathcal{E}_{tx}$) and reception ($\mathcal{E}_{rx}$) defined as follows:

$$\mathcal{E}_{tx} = B\,(\beta_1 + \beta_2\, r^{\alpha}(v,w)) \ , \tag{5.1}$$

$$\mathcal{E}_{rx} = B\,\gamma \ , \tag{5.2}$$

where α is the path loss exponent typically ranging from 2 to 6. The constants β_1, β_2, and γ correspond to the energy dissipated by the transmitter module, the transmit amplifier, and the receiver module, respectively. We denote the

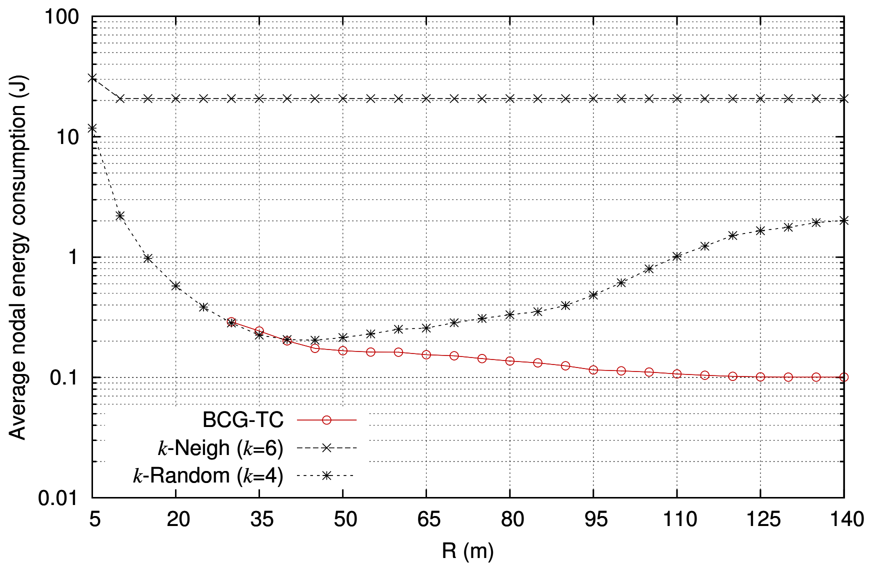


Figure 5.12: Average nodal energy consumption when the consensus has been made.

estimated distance between nodes v and w by $r(v, w)$ and the length of the message by B. We set $\alpha = 2$, equivalent to the free-space path loss model. We assume that a sensor node v can adjust its transmission power to reach its neighbors w located within the maximum radio range R. The parameters for power model used in the simulation are summarized in Table 5.2.

Using the average consensus protocol, every node exchanges (transmits and receives) its state value with its neighbors at every iteration. Thus, the total

Table 5.2: Parameters for radio model.

parameters	values
α	2
β_1	45n J/bit
β_2	10p J/bit/m^2
γ	135n J/bit
B	320 bits (40 bytes)

energy consumed by the network at the τ_{th} iteration is given by

$$\mathcal{E}_{net}(\tau) = \sum_{\substack{\forall e_{v,w} \in E \\ v \neq w}} \tau \left(\mathcal{E}_{tx} + \mathcal{E}_{rx}\right) \tag{5.3}$$

$$= \sum_{\substack{\forall e_{v,w} \in E \\ v \neq w}} \tau\, B\, \left(\beta_1 + \beta_2\, r^{\alpha}(v,w) + \gamma\right) \ . \tag{5.4}$$

If all nodes in the network reach the consensus value at the $\tau = \Omega_{\text{th}}$ iteration, then the average nodal energy consumption can be computed by $\overline{\mathcal{E}}_{node} = \mathcal{E}_{net}(\Omega)/n$. We plot $\overline{\mathcal{E}}_{node}$ in Figure 5.12. We observe that the average energy consumed at each node is determined by the average path length and the diameter of the network topologies. We also note that the nodal energy consumption for BCG-TC is consistently smaller than that of the κ-NEIGH with $\kappa = 6$ over the whole range of R.

Summary

In this chapter, we presented the BCG Topology Control protocol for densely deployed WSNs. Unlike other topology controls that are not based on specific graph models, we use a pre-defined BCG topology as a network model with which each node determines their neighbors. Note that, BCG-TC does not require nodes to memorize whole BCG topology since each node can compute its neighbors using BCG parameters and its own node ID assigned by BCG.

Moreover, we limit the nodal degree $\kappa = 4$ to reduce radio interference in BCG-TC networks. Although, it is challenging to construct a connected network with such a small nodal degree, our simulation results showed that BCG-TC can generate a large connected network with a few isolated nodes. In fact, there were only two or three isolated nodes in BCG-TC network with 1081 nodes deployed in 100m $\times$ 100m area when $R >$ 35m. We also showed that BCG-TC networks exhibit key BCG properties such as a small diameter and short average path length.

Furthermore, we were able to achieve fast information dissemination in WSNs by using BCGs as the underlying topologies of WSNs. Our performance evaluation of the information dissemination performance using the average consensus protocol revealed that we can reduce a significant amount of energy consumption for nodes to complete a collaborative job. The simulations for 1081 wireless sensor nodes distributed in a 100m $\times$ 100m area showed that BCG-TC performs well when the radio range exceeds a certain threshold and consumes the least energy among considered topology control protocols.

Chapter 6

Routing

6.1 Introduction

A routing protocol enables nodes (or routers) in a network to relay packets to a destination node (or router). The main goal of a routing protocol is to minimize the routing cost in delivering packets to a destination, where the routing cost can be defined as bandwidth (capacity), delay, or the number of hops between source and destination. It requires sophisticated methods to find the minimum cost path(s) between source and destination. For example, Internet routing protocols such as *Routing Information Protocol* (RIP) and *Open Shortest Path First* (OSPF) use Bellman-Ford algorithm [66] and Dijkstra's algorithm [67], respectively.

More recently, routing protocols for wireless sensor networks (WSNs) have gained growing attention. Wireless sensor networks consists of small battery operated nodes equipped with sensing, radio communication, and with limited computational power. In general, sensor nodes in WSNs form a distributed network without centeralized control (self-organizing network). More importantly, nodes in WSNs are frequently dying out as they consume limited energy. Due to its limited resources and ad-hoc nature, WSN requires a routing protocol that is aware of energy consumption, topology changes, network connectivity, node-to-node reachability, and efficiency of data aggregation.

As we discussed in the previous chapters, a network topology that is close to BCG can be constructed in WSNs using BCG-TC. Although it is possible to use any routing protocol in BCG networks (i.e., the network of which connections are defined by BCG), the network still requires a routing protocol that exploits the favorable properties of the constructed topology. Previously in [23], Tang and Arden proposed an iterative multi-path routing protocol for BCG that we call *Vertex Transitive BCG Routing Protocol.* However, the Vertex Transitive BCG routing protocol cannot be used in a WSN environment of which network topology keeps changing. Thus, in this chapter, we present a Dynamic BCG Routing Protocol that allows the nodes in a BCG network to update their routing table as the network topology changes over time.

6.2 Vertex Transitive BCG Routing Protocol

This section reviews the Vertex Transitive BCG Routing Protocol [23], routing table format, node ID translation, and generation method.

6.2.1 Routing table

The routing table of the Vertex Transitive BCG Routing protocol is an $(n-1) \times d$ array where n is the number of nodes, and d is a constant degree of BCGs. For example, Figure 6.1 shows an example routing table of a 4-regular BCG constructed by two distinct generators and their inverses, $\mathbb{G} = \{g_1, g_2, g_1^{-1}, g_2^{-1}\}$. Each row of the routing table is a *destination row* mapped to each destination node. The columns are *outgoing links* labeled with g_1, g_2, g_1^{-1}, and g_2^{-1} directions. Every node in the same BCG network uses the same routing table and make a forwarding decision by looking up the destination row of the routing table. The destination row and outgoing link pair of the routing table is denoted by $\mathsf{RTBL}(dst, g)$ where dst is a destination node ID, and $g \in \mathbb{G}$ is the label of an outgoing link. We assume that the blank cells are set to zero but omitted for brevity.

6.2.2 Routing table generation

The Vertex Transitive BCG routing protocol utilizes a pre-computed routing table to provide multiple shortest paths to each source and destination pair. To generate the static routing table for a BCG with n nodes and degree d, we perform the following steps:

(1) Generate a BCG, $\mathbf{C}$.

(2) Prepare a routing table RTBL with $(n-1)$ rows and d columns. Initialize RTBL with all zeros.

(3) Compute all shortest paths from a root node r (in general, $r = 0$) to all other destinations $v \in V \setminus \{r\}$. Unlike the vertex sequence of a path (Definition 2.3), generate a sequence of outgoing links for each shortest path

outgoing links

destination row	g_1	g_2	g_1^{-1}	g_2^{-1}
1	1	1		
2	1			1
3			1	
4	1		1	
5		1		
6		1	1	1
$\vdots$				
n-2			1	
n-1	1		1	1

Figure 6.1: An example of BCG ($d = 4$) static routing table for node 0. The blank cells are set to zero, but they omitted for brevity.

$\mathbf{S}_{r,v} = \{s_i \in \mathbb{G} \,|\, p_{i+1} = p_i * s_i,\ \ p_i, p_{i+1} \in P_{r,v}\}$, where $\mathbb{G} = \{g_1, g_2, g_1^{-1}, g_2^{-1}\}$, $P_{r,v}$ is the vertex sequence of a path between r and v, and $E(\mathbf{C})$ is the edge set.

(4) For each shortest path to $v \in V \setminus \{r\}$, set $\mathsf{RTBL}(v, s_1) \leftarrow 1$.

For example, consider multiple shortest paths from root node 0 to 17 as follows:

$$\text{Path 1: } 0 \xrightarrow{g_1} 3 \xrightarrow{g_1} 6 \xrightarrow{g_2^{-1}} 17 \ ,$$

$$\text{Path 2: } 0 \xrightarrow{g_2} 4 \xrightarrow{g_1} 10 \xrightarrow{g_2} 17 \ ,$$

$$\text{Path 3: } 0 \xrightarrow{g_1^{-1}} 18 \xrightarrow{g_2^{-1}} 8 \xrightarrow{g_1^{-1}} 17 \ .$$

Then, the sequences of the shortest paths are

$$\text{Path 1: } S_{0,17} = \{g_1\,, g_1\,, g_2^{-1}\} \ ,$$
$$\text{Path 2: } S_{0,17} = \{g_2\,, g_1\,, g_2\,\} \ ,$$
$$\text{Path 3: } S_{0,17} = \{g_1^{-1}, g_2^{-1}, g_1^{-1}\} \ .$$

For each shortest path to the destination 17, set the routing table as follows:

$$\begin{aligned}
&\text{Path 1: } \mathsf{RTBL}(17, g_1\ ) \leftarrow 1\ \ , \\
&\text{Path 2: } \mathsf{RTBL}(17, g_2\ ) \leftarrow 1\ \ , \\
&\text{Path 3: } \mathsf{RTBL}(17, g_1^{-1}) \leftarrow 1\ \ .
\end{aligned}$$

which yields the destination row 17 for the routing table as shown in Figure 6.2.

	g_1	g_2	g_1^{-1}	g_2^{-1}
⋮				
17	1	1	1	
⋮				

Figure 6.2: Example routing table with the destination row 17 set based on the shortest paths.

If node 0 receives a packet destined to 17, then it looks up destination row 17 of the routing table and figures out the outgoing links that guarantee the shortest path to 17. In this case, there are three available outgoing links g_1, g_2, and g_1^{-1}. Since there are multiple available outgoing links, node 0 randomly selects one link among them for forwarding.

6.2.3 Node ID translation

Recall that BCGs are vertex-transitive graphs, that is, the topology viewed from any node looks identical (see Definition 2.9). This vertex-transitive property makes it possible for every node in the same BCG to use the same routing table that is pre-computed for one root node (in general, root node ID r=0). Once the routing table is stored, each node translates a destination node ID into a new node ID which is the destination node ID viewed from the root node. Thus, every node can view the routing table as if it were the root node.

Following the node ID translation in [23], let i be the ID of a node making a routing decision, and let j be the destination node ID extracted from a packet to be forwarded. If a BCG is generated by a parameters p, k, and a, then the translated destination node ID j' is computed as follows:

$$j' = \left\langle a^{q-c_1}\left(m_2 - m_1\right)\right\rangle_p q + \left\langle c_2 - c_1\right\rangle_q \ , \tag{6.1}$$

where $\langle\rangle_x$ is the x-modulo operator, $m_1 = i/q$, $m_2 = j/q$, $c_1 = \langle i\rangle_q$, and $c_2 = \langle j\rangle_q$ are integers.

For example, suppose node 7 on a 21 nodes BCG with parameter $p = 7$, $q = k = 3$, and $a = 2$ receives a packet destined to node 17. Then, plugging $m_1 = 7/3 = 2$, $m_2 = 17/3 = 5$, $c_1 = \langle 7 \rangle_3 = 1$, and $c_2 = \langle 17 \rangle_3 = 2$ into Eq. (6.1) yields the new destination node ID $j' = 16$. Thus, node 7 looks up the row 16 of its routing table to forward the packet destined to 17.

6.3 Dynamic BCG Routing Protocol

This section presents the Dynamic BCG Routing Protocol for each node to update its routing table as topology changes dynamically. Each node updates its routing table as soon as the topology changes to maximize the routing efficiency and prevent packet losses. In addition, we propose a Backward Advertisement to propagate topology changes to the neighbors of a failed node.

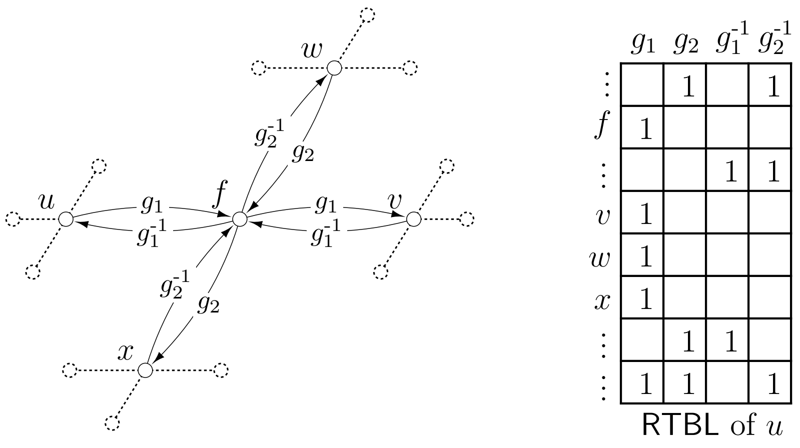
6.3.1 Assumption

Densely deployed wireless networks. We assume that the wireless network under our consideration is very dense and that each node's radio range is sufficiently large to communicate with its logical neighbors defined by BCG. The communication range and topology control issues for densely deployed wireless sensor nodes have been discussed in Chapter 5.

Link status detection. Even though we do not specify a link state detection method, we assume a node can detect the status of links (e.g., *link up* and *link down*) to its immediate neighbors. In general, link state can be monitored by sending hello packets and collecting feedback from neighboring nodes. A node or its links may fail either temporarily or permanently, however, we consider a scenario where nodes are dying out permanently due to energy depletion.

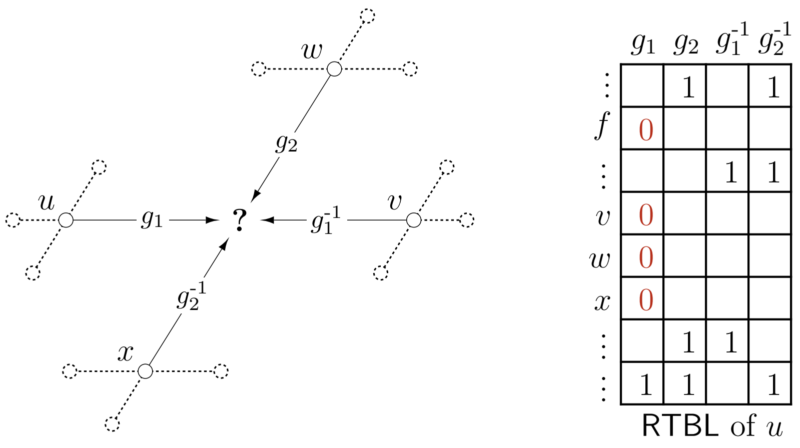
Dynamically changing topology. We assume the topology of a network to change over time. More specifically, we consider a wireless sensor network where nodes die out as they consume limited energy while operating various functions such as sensing, processing, transmitting, and receiving data.

Routing control time. The *routing control time* includes (i) link down detection time, (ii) routing table update time, and (iii) routing control message building, sending, reception, and processing time. We assume that the routing control time after each node failure is no greater than the time between node failures.



	g_1	g_2	g_1^{-1}	g_2^{-1}
⋮		1		1
f	1			
⋮			1	1
v	1			
w	1			
x	1			
⋮		1	1	
⋮	1	1		1

(a) Topology and the routing table of u before f fails.



	g_1	g_2	g_1^{-1}	g_2^{-1}
⋮		1		1
f	0			
⋮			1	1
v	0			
w	0			
x	0			
⋮		1	1	
⋮	1	1		1

(b) Topology and the routing table of u after f fails.

Figure 6.3: Illustration of destination blocking after node f fails using the Dynamic BCG Routing Protocol. Note that some of the original edges of nodes (depicted in dotted line) are omitted for brevity.

Routing table. Every node in the network starts with an identical but pre-translated routing table using Eq. (6.1). So that nodes do not need to translate the destination node ID at runtime. Unlike the Vertex Transitive BCG Routing Protocol, as the network topology changes over time, the routing tables of the Dynamic BCG Routing Protocol will be independently updated by nodes and may no longer be identical with one another.

6.3.2 Dynamic routing table update

As nodes fail, the neighbors of failed node detect the link state change, and update their routing table accordingly. In Figure 6.3(a), we illustrate the part of a BCG where 5 nodes are connected by four generators $\mathbb{G} = \{g_1, g_2, g_1^{-1}, g_2^{-1}\}$. The figure also shows a sample routing table for node u. The routing table and the topology in Figure 6.3(a) show that u will forward packets destined to f, w, x, or v to its g_1-direction outgoing link. Although the routing table of f is not presented in the figure, we can guess that f will forward packets destined to its immediate neighbors u, v, w, and x respectively to its g_1^{-1}, g_1, g_2^{-1}, and g_2 outgoing link.

Now, suppose a node f in Figure 6.3(a) fails, and its immediate neighbors $\mathcal{N}_f = \{u, v, w, x\}$ detect that f is not responding. Since forwarding packets to the failed node f is undesirable, neighbors in $\mathcal{N}_f$ block outgoing links to f by updating their routing table. In addition, each node in $\mathcal{N}_f$ also blocks the nodes in $\mathcal{N}_f$ excluding itself. Thus, neighbors of f block destinations as follows:

$$\begin{aligned}
\text{node } u\text{:}\ & \mathsf{RTBL}(\{f, v, w, x\}, g_1) \leftarrow 0\ , \\
\text{node } v\text{:}\ & \mathsf{RTBL}(\{f, u, w, x\}, g_1^{-1}) \leftarrow 0\ , \\
\text{node } w\text{:}\ & \mathsf{RTBL}(\{f, u, v, x\}, g_2) \leftarrow 0\ , \\
\text{node } x\text{:}\ & \mathsf{RTBL}(\{f, u, v, w\}, g_2^{-1}) \leftarrow 0\ .
\end{aligned}$$

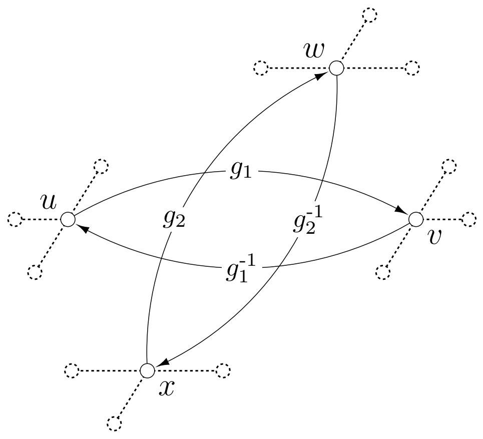
The resulting routing table of u, after destination blocking, is shown in Figure 6.3(b).

6.3.3 Dynamic routing table update with CTR

The CTR algorithm presented in Chapter 4 ensures a BCG to remain connected after pruning a large amount of nodes from it. In fact, 98.9 $\sim$ 100% pruned BCGs were connected even after 80% size reduction, depending on the original BCG size. Thus, in addition to the dynamic routing table update described in the previous section, we also update the routing table according to the new neighbors found by the CTR algorithm.

If a node detects a link down, then it performs the dynamic routing table update followed by CTR to maintain network connectivity. In addition, we assume that nodes exchange the list of their immediate neighbors *nbrlist* with their newly found neighbors. These neighbor lists will be used by the Backward Advertisement algorithm described in Section 6.4 to improve the reachability.

Once a node performed CTR and found a new neighbor, it updates its routing table according to a new connection established by CTR. Let us con-



	g_1	g_2	g_1^{-1}	g_2^{-1}
⋮		1		1
f	0			
⋮			1	1
v	1			
w	0			
x	0			
⋮		1	1	
⋮	1	1		1

RTBL of u

Figure 6.4: Illustration of topology and routing table of u after CTR. Note that some of the original edges of nodes (depicted in dotted line) are omitted for brevity.

sider CTR algorithm performed on the partial BCG topology with failed node f in Figure 6.3(b). After executing CTR, nodes u and v are rewired by the edges in g_1 and g_1^{-1} directions, while nodes w and x are joined by edges in g_2 and g_2^{-1} directions. Next, nodes in $\mathcal{N}_f$ connected to new neighbors also update their routing table as follows:

$$\begin{aligned}
\text{node } u\text{: } & \mathsf{RTBL}(v, g_1) \leftarrow 1 \ , \\
\text{node } v\text{: } & \mathsf{RTBL}(u, g_1^{-1}) \leftarrow 1 \ , \\
\text{node } w\text{: } & \mathsf{RTBL}(x, g_2^{-1}) \leftarrow 1 \ , \\
\text{node } x\text{: } & \mathsf{RTBL}(w, g_2) \leftarrow 1 \ .
\end{aligned}$$

Let $\mathcal{N}_x$ be a set of neighbors of node x, then each node exchanges its list of neighbors with its newly found neighbor. For example, node u in Figure 6.4 receives the list of $\mathcal{N}_v$ from its new neighbor v with which u updates its routing table. The routing table updates performed by the neighbors of the failed node are as follows:

$$\begin{aligned}
\text{node } u\text{: } & \mathsf{RTBL}(\mathcal{N}_v \setminus \{u\}, g_1) \leftarrow 1 \ , \\
\text{node } v\text{: } & \mathsf{RTBL}(\mathcal{N}_u \setminus \{v\}, g_1^{-1}) \leftarrow 1 \ , \\
\text{node } w\text{: } & \mathsf{RTBL}(\mathcal{N}_x \setminus \{w\}, g_2^{-1}) \leftarrow 1 \ , \\
\text{node } x\text{: } & \mathsf{RTBL}(\mathcal{N}_w \setminus \{x\}, g_2) \leftarrow 1 \ .
\end{aligned}$$

Figure 6.4 show the updated routing table of node u.

Table 6.1: Contents of Backward Advertisement packet.

Field	Description
destid	destination (BA packet receiver) node ID
srcid	source (BA packet sender) node ID
blkdest	ID of failed node
nbrlist	a list of new neighbors

6.3.4 Random forwarding

As packets arrive at a node, it looks up its routing table to determine the best outgoing link(s) to forward the packet (i.e., 1's in the specific destination row of the routing table). If there are multiple outgoing links available (i.e., multiple 1's in the same row), then the node randomly picks one outgoing link among them.

However, a node may not be able to decide the best outgoing link to the destination by looking up its routing table. This is because some destination rows may become all zeros as the topology changes with time. To deal with this case, we use random forwarding.

Suppose node u in Figure 6.4 receives a packet destined to x. The destination row x of u's routing table contains all zeros. Recall that blank cells in the figure are assumed to be set to zero but omitted for brevity. Since u is not able to find the best outgoing link to destination x, it randomly selects the outgoing link towards one of its neighbors $\mathcal{N}_u \setminus \{s\}$, where s is the node that sent the packet to u. We exclude s to avoid infinite routing loops.

We concede this random forwarding may cause packets to (i) traverse a network infinitely or (ii) take unnecessarily long paths to reach a destination. To reduce the frequency of random forwarding, we propose the *Backward Advertisement* (BA) algorithm in the next section.

6.4 Backward Advertisement

The Backward Advertisement algorithm prevents packets from being delivered to nodes that do not have an optimal path to the destination. Instead, BA algorithm gives its neighboring nodes a chance to take the packet forwarding responsibility. Because the BCG routing table supports multiple shortest paths between nodes, those neighboring nodes may have one or more alternative shortest paths to the destination (i.e., 1's in the specific destination row of the routing table). In case nodes who received BA packets do not have the shortest

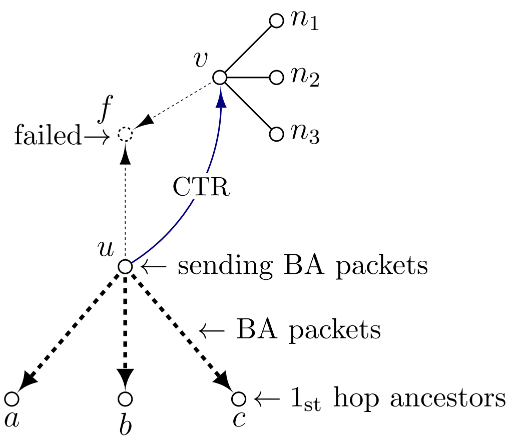


Figure 6.5: Illustration of u forwarding BA packets to its first hop ancestors as it detects a link down for f. Then, the first hop ancestors send new BA packets to their own ancestors which are the second order of u. Note that some nodes and edges are not shown for clarity.

path either, they can propagate BA packets one hop further to their neighbors. However, to limit the number of BA packets generated in the network, we set the number of hops to propagate BA packets to within two hops from a failed node. The heuristics behind this restriction will be discussed in Section 6.6.

6.4.1 Operation

The BA algorithm uses a simple control message called BA packet. The main contents of a BA packet is the list of blocked destinations including the failed node and the list of new neighbors found by CTR. Using the BA packet, a node informs its neighbors of the most recent changes such as a node failure. In addition to informing node failure, BA also lets its neighboring nodes update their routing table of newly connected neighbors. More specifically, if a node found a new neighbor by performing CTR, then it updates its routing table with 1 for the destination row corresponding to the new neighbor. Next, it sends BA packets to its neighboring nodes to have them update their routing tables with 1 for each new neighbor discovered by CTR. The BA packet carries the information summarized in Table 6.1.

The BA operation can only be initiated by nodes that detect node failures. When a node detects a link down, it updates its routing table according to the routing table update method described in Section 6.3.2 and Section 6.3.3. Next, it initiates the BA operation by building a BA packet and sends the BA packet to its first hop neighbors as shown in Figure 6.5. Finally, the neighbors update their routing table using the information stored in BA packets.

Let us consider Figure 6.5 that shows how BA packets are propagated in the network. In the figure, u detects the link connected to f is down and initiates the BA operation by building and sending BA packets to its neighbors depicted in Figure 6.6.

$destid = a$	$srcid = u$	$blkdest = f$	$nbrlist = \{n_1, n_2, n_3\}$
$destid = b$	$srcid = u$	$blkdest = f$	$nbrlist = \{n_1, n_2, n_3\}$
$destid = c$	$srcid = u$	$blkdest = f$	$nbrlist = \{n_1, n_2, n_3\}$

Figure 6.6: Node u forwards BA packets to its neighbors a, b, and c illustrated in Figure 6.5.

Upon receiving the BA packet, the neighbors (i.e., BA packet recipients) extract the destination node ID *destid* (a, b, or c), the blocked destination ID *blkdest* (f), and a new neighbor list *nbrlist* ($\{n_1, n_2, n_3\}$) from the BA packet. Then, each BA packet receiver checks if *destid* matches its ID. If it does not match, the BA packet is discarded. Otherwise, the BA receiver records the BA packet incoming link g, blocks the destination by performing $\mathsf{RTBL}(blkdest, g^{-1}) \leftarrow 0$, and updates its routing table by setting $\mathsf{RTBL}(nbrlist, g^{-1}) \leftarrow 1$.

6.5 Routing Performance

6.5.1 Setups

We generated the original BCGs with two different sizes $n = 506$ and 1081. Then, we performed BCG pruning with CTR to simulate node failures. For each node removal (failure), the proposed Dynamic BCG Routing Protocol has been performed. Also, for each amount of node failures (i.e., 10%, 20%, . . . 80% failures), we evaluated the routing performance in terms of reachability and average hop count. For each amount of node removals, we generated 99 samples and obtained the average of performance metrics considered.

6.5.2 Reachability

We define *reachability* η as the number of reachable source and destination pairs among all pairs of nodes in a network. To determine the reachable source-destination pairs, we set the time-to-live (TTL) [1] to the size of the

[1] The idea of TTL is borrowed from the Internet Protocol. The TTL limits the number of hops that a packet can traverse in a network. For example, the TTL is set to a pre-defined

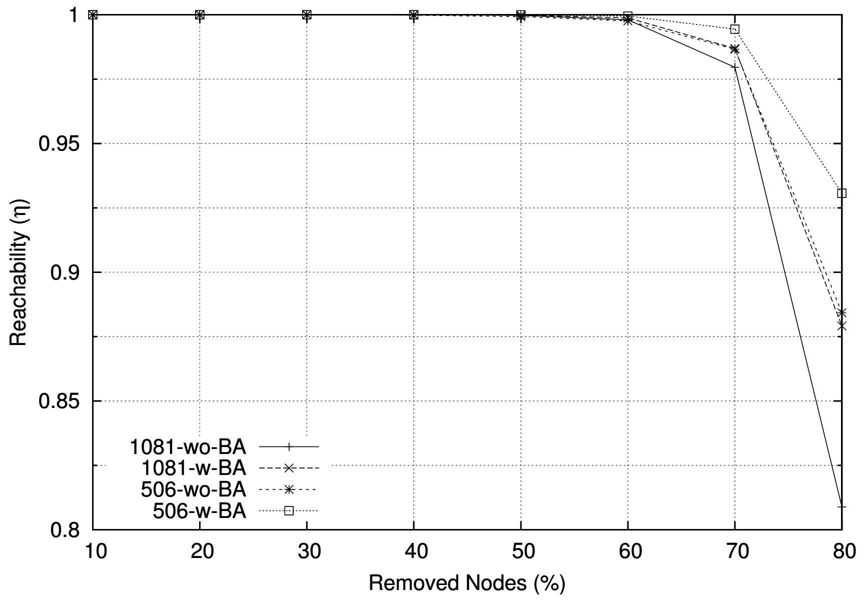


Figure 6.7: Reachability versus the percentage of node failures.

original BCGs. If any packets traverse nodes more than a pre-defined TTL value before reaching a destination, then we assume that the destination is unreachable, and we discard the packet.

We evaluated the node to node reachability (η) as a function of the number of failed nodes. This measures how effectively the proposed routing protocol finds a routing path between a source and destination. We obtained the reachability by computing the ratio of the number of reachable pairs of nodes n_r to all possible pairs of nodes alive n_a, that is, $\eta = 2\,n_r/\,(n_a\ (n_a - 1))$.

Figure 6.7 shows the reachability as a function of the percentage of node failures. For each network sample, Dynamic BCG Routing Protocol with and without Backward Advertisement has been simulated (wo-BA and w-BA in the figure). Beyond 70% node failures, the BA algorithm starts to improves the reachability comparing to the Dynamic BCG Routing protocol without it. However, BA improves the reachability by 5% and 7% when 80% of nodes are failed from the $n = 506$ and $n = 1081$ networks, respectively. Note that, this improvement with BA may seem to be minimal but 7% improvement for a BCG with $n = 1081$ and 80% failed nodes (i.e., a network with 216 nodes) is equivalent to 1625 more reachable source and destination pairs.

value when a packet is generated and reduced by one on every hop. If any intermediate node receives a packet with TTL value equals zero, then it discards the packet.

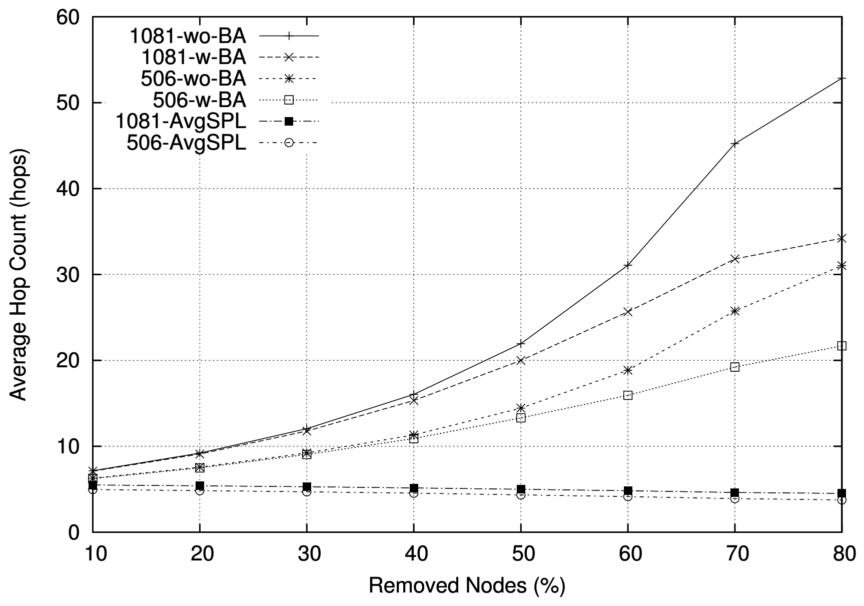


Figure 6.8: Average hop count and average shortest path length versus the percentage of node failures.

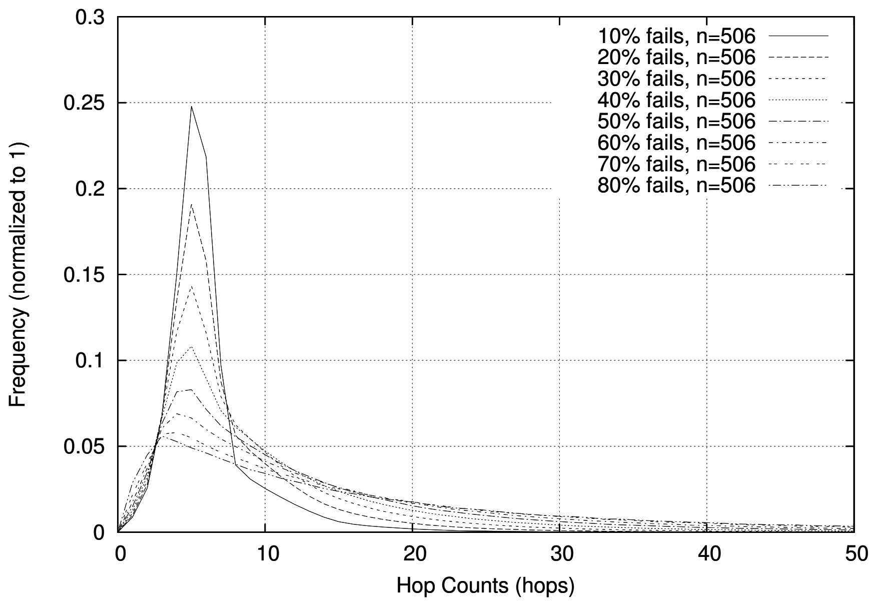
6.5.3 Average hop count

We computed the average of the average hop counts of the 99 network samples for each node removals. In addition, we computed the average shortest path length which serves as a reference representing the best achievable average hop count for each network sample.

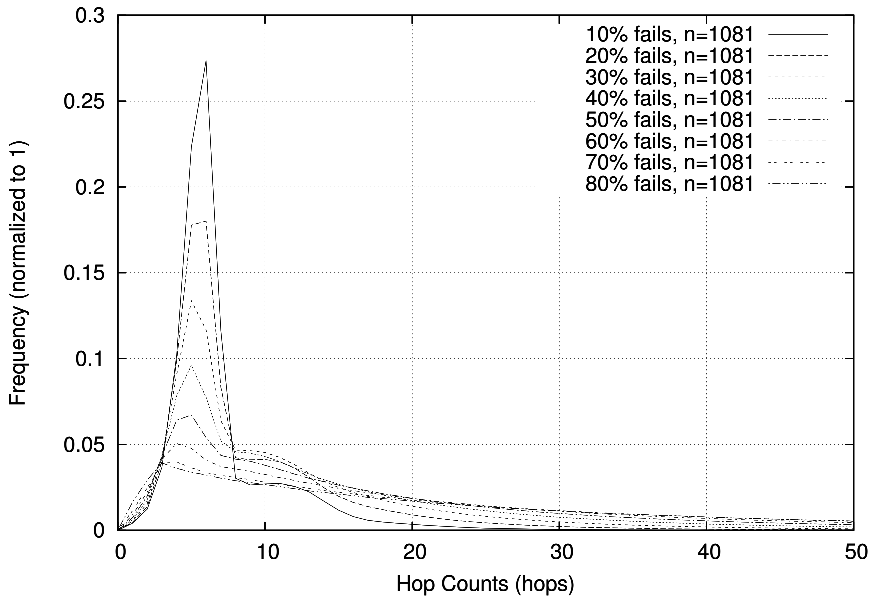
Figure 6.8 shows the results. The gap between the lower bound of the hop counts represented in the average shortest path length (AvgSPL in the figure) and the average hop count using the proposed routing table update method keeps widening with the number of failed nodes. However, we observed that BA reduces the average hop count up to 35% from that of the Dynamic BCG Routing Protocol without BA.

6.5.4 Hop counts distribution

We further investigated the distribution of hop count when using the Dynamic BCG Routing Protocol with BA. In Figure 6.9, hop count histograms are shown for BCGs with $n = 506$ and 1081. The hop counts for every source and destination pair were collected and their occurrence (normalized to one). As expected, the occurrences of short routing paths decrease while those of long routing paths increase with the number of failed nodes. Thus, minimizing the



(a) $n = 506$



(b) $n = 1081$

Figure 6.9: Hop count distribution for the Dynamic BCG Routing Protocol with Backward Advertisement. Note that Hop counts exceeding 50 are not shown for brevity.

occurrences of such long routing paths will be the main goal to improve the proposed routing protocol, which remains as future work.

In addition, modes of the histograms are found between $3 \sim 5$ hops and $3 \sim 6$ hops for $n = 506$ and $n = 1081$, respectively. These almost match the average shortest path lengths (AvgSPL) in Figure 6.8 which span between $3.75 \sim 4.97$ and $4.52 \sim 5.49$, respectively for the BCGs with $n = 506$ and $n = 1081$.

6.6 Discussion

In the previous sections, we observed that BA can reduce the average hop count and improve the node to node reachability. Currently, we only propagate BA packets to a failed node's neighbors within two hops. The BA can be used more aggressively to minimize the average hop count, while this increases the BA packets traveling around the network for every node failure. In fact, the number of BA packets propagated to the network per node failure is a function of the BA packet propagation range that is the number of hops from the failed node. Let $h > 1$ be the BA packet propagation range and let η_h be the number of BA packets propagated. Assume that each node within h hops from a failed node is unique node having d neighbors, then

$$\begin{aligned}
\eta_2 &= d\,(d-1)\ , \\
\eta_3 &= (d-1)\,\eta_2 = d\,(d-1)^2\ , \\
\eta_4 &= (d-1)\,\eta_3 = d\,(d-1)^3\ , \\
&\vdots
\end{aligned}$$

Thus, the number of BA packets required to be propagated is given by $\eta_h = d\,(d-1)^{h-1}$. For example, if the network is 4 regular, then the number of BA packets to be generated for each node failure, $\eta_3 = 36$, $\eta_4 = 108$, $\eta_4 = 324$, and so on, which increase quickly with h. Moreover, only 10% of nodes failing from a 1081 nodes 4-regular BCG network will result in total $108 \times \eta_2 = 1296$, $108 \times \eta_3 = 3888$, $108 \times \eta_4 = 11664$ BA packets propagated in the network. Energy required for wireless sensor nodes to generate, transmit, receive, and process a large number of BA packets will be a huge burden to the nodes suffering from limited energy source.

Summary

This chapter presents the Dynamic BCG Routing Protocol for a BCG topology that is dynamically changing over time. We also proposed the Backward Advertisement that allows nodes to propagate topological changes to their neighbors. The results in our simulation setup showed that BA improves the average hop count and the reachability of BCG Dynamic Routing Protocol up to 7% and 35%.

Chapter 7

Conclusion

In this thesis, we investigated a graph theoretic approach to achieve ultrafast information dissemination and data exchange in large and dense networks. Particularly, we focused on the strengths, limitations and solutions, and applications of Borel Cayley Graph (BCG) that is a regular, vertex transitive, and algebraically generated pseudo-random graphs. We summarize our contributions and provide our insights on the future research topics and directions.

7.1 Summary

Network model for ultrafast information dissemination. Among many well known network models including regular ring lattices, random graphs, toroidal meshes, diagonal meshes, and small-world networks, we confirmed that BCGs are one of the most favorable network models. In fact, our comparative study showed that BCGs have superior properties including deterministic topology generation, small nodal degree, short average path length, small diameter, and ultrafast information dissemination.

Resolving size limitation of BCGs The BCG's inflexible sizes has posed a severe limitation on applying this superior network model in real networks. We resolved this size restriction by using proposed BCG Pruning and BCG Random Expansion algorithms. Analytical analysis and extensive simulations confirmed that the original BCGs and the Quasi BCGs exhibit almost the same information dissemination performance as well as topological properties.

BCG Topology Control for Wireless Sensor Networks To construct a BCG topology in Wireless Sensor Network (WSN), we developed BCG Topology Control (BCG-TC). We compare the performance of BCG-TC networks to κ-NEIGH and κ-Random in terms of topological metrics including a diameter, average path length, network connectivity, and energy consumption. Particularly with a challengingly small maximum nodal degree $k \leq 4$, the BCG-TC constructed a giant component containing more than 98% of total nodes when maximum radio range R exceeds approximately 25% of diagonal length of a rectangular area. Moreover, comparison on energy consumption showed nodes in BCG-TC networks to consume the least energy among the topology control protocols considered.

BCG Dynamic Routing Protocol We proposed *Dynamic BCG Routing Protocol* based on the BCG's original routing protocol. The Dynamic BCG Routing Protocol allows every node in a network dynamically to update their routing table using the proposed routing protocol as nodes die out over time. The proposed routing protocol utilizes our Backward Advertisement (BA) to propagate topological changes to nearby neighbors. The results in our simulation setup showed that BA improves the average hop counts and the reachability of BCG Dynamic Routing Protocol up to 7% and 35%.

7.2 Future Work

7.2.1 Open issues

Measuring the randomness of a graph. In our research, we realized that it is not easy to define or measure the randomness of a graph. One may want to define randomness by means of degree distribution, however, graphs display the same degree distribution such as k-regular ring lattices and BCGs have totally different characteristics. Other would like to use statistics or graph metrics to define the graph randomness, yet many different graphs may share the same quantity. Thus, answering to the question, "How random a random graph is?" or "How to measure the randomness of regular graphs?" will be challenging. However, we expect the correct measure of graph randomness can answer its relationship to other graph metrics or performance measures.

Selecting right BCG parameters. The diameter, average path length, information dissemination performance, and even the connectivity of BCGs are determined by the choice of BCG parameters. Although choosing a right set of BCG parameters is important, there has been lack of a comprehensive study on the choice of BCG parameters. In this regard, we have provided some heuristic guidelines on selecting BCG generators in Section 3.6. However, even though the guidelines ensure connected BCGs, we are still missing a mathematically proven method to obtain right BCG parameters generating connected BCGs with better topological properties. We expect the investigation on obtaining right BCG parameters will give us more insights into the properties and applications of BCGs.

Deterministic BCG expansion. The proposed BCG Random expansion enlarges the original BCGs in any larger sizes. However, unlike the BCG Pruning algorithm that uses BCG's algebraic connection rule, the BCG Expansion algorithm randomly rewires connections between added nodes and existing nodes.

Pruning and expanding other network models. Beyond application to the BCGs we expect some algorithms similar to our BCG Pruning and Expansion algorithm may be able to resize other network models such as Ramanujan graphs [68] or de Bruijn graphs [69]. It will be interesting to investigate the topological and spectral properties, as well as the information dissemination performance of such resized graphs. We concede that this random rewiring would prevent us from using the BCG's useful properties such as algebraically defined connections between nodes. In fact, the core of BCG Pruning algorithm is the CTR algorithm that utilizes the BCG's algebraic connection rule. Thus, we expect that BCG's connection rule might be used to deterministically expand the BCGs.

7.2.2 Applications

Large clusters processing peta-scale data. In 2008, Dean et al. [7] reported that Google's large-scale cluster processes twenty petabytes (10^{15} bytes) of data per day to serve its users. Same year, Baeza-Yatesand et al. [70] reported Yahoo!'s clusters handle dataset of multi-petabyte order. Those clusters or server-farm usually consists of several thousands of networked machines where each machine own or share multiple, several terabytes hard-disks. In these very dense and data-intensive networked machine environments, not only the physical connections among servers, but also the logical connections between datasets are key to get desired jobs done on time.

We believe that the BCGs and the Quasi BCGs are strong candidates for connecting those physical or logical objects. For instance, let us consider a

4-regular BCG[1] with $15,657$ nodes whose diameter is only 10. If a cluster of approximately 15000 servers[2] (or networked storages) are connected by Quasi BCG topology, then any servers in the cluster can reach any other servers (at most) in 10 hops. The algebraic connection of BCGs will be a key benefit for such large scale clusters as every server can compute (instead of being configured) its neighbors to connect using very simple algebraic operation. Moreover, we believe our CTR algorithm provide fault tolerant feature too. Even if a large number of servers fail (e.g., up to 80%), our CTR algorithm will guarantee a strong connectivity for the rest of servers alive (see connectivity analysis in Section 4.3.1). In this massive failure scenario, CTR allows each servers to find its next available neighbors independently and deterministically without human intervention.

Mobile ad hoc networks. In mobile ad hoc networks, nodes are moving within the coverage of base station (BS) or between the BSs. As traveling between BSs, nodes are registered to one BS and unregistered from another BS, which requires remaining nodes (as opposed to moving nodes) to adapt to frequent topology changes. In this scenario, our BCG Pruning and BCG Random Expansion might be used to maintain connectivity of remaining nodes in the network. Specifically, when nodes are moving out of radio coverage, remaining nodes who lost connections to the moving nodes uses BCG Pruning to maintain their connectivity. When new nodes appears in the network, nearby neighbors may be able to use BCG Expansion to give connections to the new nodes. It is also a possible scenario that the centralized controllers or the BSs perform BCG Pruning and Expansion and manage connections of mobiles stations under coverage.

[1] Note that the nodal degree is not limited to four, but rather we can use any even nodal degree by using different set of generators. Intuitively, BCGs with larger nodal degree will result in smaller diameter, shorter average path length, and better information dissemination performance.

[2] It does not need to be exactly $15,657$ as the original BCGs can be downsized using BCG Pruning algorithm.

Bibliography

[1] Lakshminarayanan Subramanian, Ion Stoica, Hari Balakrishnan, and Randy H. Katz. Overqos: offering internet qos using overlays. *SIGCOMM Comput. Commun. Rev.*, 33:11–16, January 2003. ISSN 0146-4833.

[2] M. Castro, P. Druschel, Y.C. Hu, and A. Rowstron. Topology-aware routing in structured peer-to-peer overlay networks. In *Future Directions in Distributed Computing*, pages 103–107. Springer-Verlag, 2003. ISBN 3540009124.

[3] Jeremy Elson and Kay Römer. Wireless sensor networks: a new regime for time synchronization. *SIGCOMM Comput. Commun. Rev.*, 33:149–154, January 2003. ISSN 0146-4833.

[4] P.S. Dodds and D.J. Watts. A generalized model of social and biological contagion. *Journal of Theoretical Biology*, 232(4):587–604, 2005. ISSN 0022-5193.

[5] D. López-Pintado. Diffusion in complex social networks. *Games and Economic Behavior*, 62(2):573–590, 2008. ISSN 0899-8256.

[6] Meeyoung Cha, Alan Mislove, and Krishna P. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 721–730, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4.

[7] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008. ISSN 0001-0782.

[8] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 63–74. ACM, 2008.

[9] G. Lee, N. Tolia, P. Ranganathan, and R.H. Katz. Topology-aware resource allocation for data-intensive workloads. In *Proceedings of the first ACM asia-pacific workshop on Workshop on systems*, pages 1–6. ACM, 2010.

[10] D. J. Watts and S. H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393(6684):440–442, June 4, 1998. ISSN 0028-0836.

[11] R. Olfati-Saber. Ultrafast consensus in small-world networks. In *Proceedings of the American Control Conference (ACC'05)*, volume 4, pages 2371–2378, June 8–10, 2005.

[12] R. Olfati-Saber. Algebraic connectivity ratio of Ramanujan graphs. In *Proceedings of the American Control Conference (ACC'07)*, pages 4619–4624, July 9–13, 2007.

[13] K. Wendy Tang and B. W. Arden. Representations of Borel Cayley graphs. *SIAM Journal on Discrete Mathematics*, 6(4):655–676, 1993. ISSN 0895-4801.

[14] Chalermek Intanagonwiwat, Deborah Estrin, Ramesh Govindan, and John Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, pages 457–458, 2002. ISBN 0-7695-1585-1.

[15] A. Hu and S.D. Servetto. Asymptotically optimal time synchronization in dense sensor networks. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 1–10. ACM, 2003.

[16] I. Demirkol, C. Ersoy, and F. Alagoz. MAC protocols for wireless sensor networks: a survey. *IEEE Commun. Mag.*, 44(4):115–121, April 2006.

[17] Kirk Martinez, Jane K. Hart, and Royan Ong. Environmental sensor networks. *Computer*, 37(8):50–56, 2004. ISSN 0018-9162.

[18] S. Coleri, S.Y. Cheung, and P. Varaiya. Sensor networks for monitoring traffic. In *Proceedings of the Allerton conference on communication, control and computing*, pages 32–40, 2004.

[19] Ting Yan, Tian He, and John A. Stankovic. Differentiated surveillance for sensor networks. In *Proceedings of the First international conference on Embedded networked sensor systems (SenSys'03)*, pages 51–62. ACM, 2003. ISBN 1-58113-707-9.

[20] Paolo Santi. Topology control in wireless ad hoc and sensor networks. *ACM Computing Surveys (CSUR)*, 37(2):164–194, 2005. ISSN 0360-0300.

[21] D.M. Blough, M. Leoncini, G. Resta, and P. Santi. The k-neighbors approach to interference bounded and symmetric topology control in ad hoc networks. *IEEE Trans. Mobile Comput.*, 5(9):1267–1282, 2006.

[22] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA'99. Second IEEE Workshop on*, pages 90–100. IEEE, 2002. ISBN 0769500250.

[23] K. Wendy Tang and B. W. Arden. Vertex-transitivity and routing for Cayley graphs in GCR representations. In *Proceedings of the ACM/SIGAPP Symposium on Applied Computing (SAC'92)*, pages 1180–1187, New York, NY, USA, 1992. ACM. ISBN 0-89791-502-X.

[24] Fatihcan M. Atay, Tuerker Biyikoglu, and Juergen Jost. Synchronization of networks with prescribed degree distributions. *IEEE Trans. Circuits Syst. I*, 53:92, 2006.

[25] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286:509–512, October 1999.

[26] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the Internet topology. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication (SIGCOMM'99)*, pages 251–262, New York, New York, 1999. ACM. ISBN 1-58113-135-6.

[27] Sven Bilke and Carsten Peterson. Topological properties of citation and metabolic networks. *Phys. Rev. E*, 64:036106, 2001.

[28] M. E. J. Newman. Scientific collaboration networks. I. Network construction and fundamental results. *Phys. Rev. E*, 64(1):016131, June 2001.

[29] M. E. J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, January 2001.

[30] Walter W. Powell, Douglas R. White, Kenneth W. Koput, and Jason Owen-Smith. Network dynamics and field evolution: The growth of interorganizational collaboration in the life sciences. *American Journal of Sociology*, 110(4):1132–1205, 2005.

[31] M. E. J. Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter*, 38(2):321–330, 2004.

[32] A. Jamakovic, S. Uhlig, and I. Theisler. On the relationships between topological metrics in real-world networks. In *Proceedings of the European Conference on Complex Systems*, pages 1–14, 2007.

[33] A. Banerjee. *The Spectrum of the graph Laplacian as a tool for analyzing structure and evolution of networks.* PhD thesis, Max Planck Institute, Berlin, Germany, 2008.

[34] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 1973.

[35] B. Mohar. The Laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, 2:871–898, 1991.

[36] F.R.K. Chung. *Spectral graph theory.* American Mathematical Society, 1997.

[37] C.D. Godsil and G. Royle. *Algebraic graph theory.* Springer, 2001.

[38] C.D. Meyer. *Matrix analysis and applied linear algebra.* Society for Industrial and Applied Mathematics, 2000.

[39] S. White and P. Smyth. A spectral clustering approach to finding communities in graph. In *Proceedings of the SIAM Data Mining Conference*, 2005.

[40] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. In *Proceedings of the ACM/IEEE conference on Supercomputing*, volume 95, page 285, New York, NY, USA, 1995. ACM.

[41] C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3d meshes. *ACM Transactions on Graphics*, 22(3):358–363, 2003.

[42] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. Autom. Control*, 49(9):1520–1533, September 2004. ISSN 0018-9286.

[43] P. Erdös and A. Rényi. On the evolution of random graphs. *Bulletin of the Institute of International Statistics*, 38:343–347, 1961.

[44] F. Comellas and S. Gago. Synchronizability of complex networks. *J. Phys. A: Math. Theor*, 40:4483–4492, 2007.

[45] H. Hong, Beom Jun Kim, M. Y. Choi, and Hyunggyu Park. Factors that predict better synchronizability on complex networks. *Phys. Rev. E*, 69 (6):067105, June 2004.

[46] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks (IPSN'05)*, pages 63–70, 2005. ISBN 0-7803-9202-7.

[47] Y. Hatano and M. Mesbahi. Agreement over random networks. *IEEE Trans. Autom. Control*, 50(11):1867–1872, November 2005. ISSN 0018-9286.

[48] R. Olfati-Saber and J. S. Shamma. Consensus filters for sensor networks and distributed sensor fusion. In *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference*, pages 6698–6703, December 2005.

[49] J.A. Fax and R.M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Trans. Autom. Control*, 49(9):1465–1476, September 2004. ISSN 0018-9286.

[50] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobilerobots with limited visibility. *IEEE Trans. Robot. Autom.*, 15(5):818–828, 1999.

[51] J. Lin, AS Morse, and BDO Anderson. The multi-agent rendezvous problem. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, volume 2, pages 1508–1513, 2003.

[52] J. Cortes, S. Martinez, and F. Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Trans. Autom. Control*, 51(8):1289–1298, 2006.

[53] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proc. IEEE*, 95(1):215–233, January 2007. ISSN 0018-9219.

[54] S.H. Strogatz. Exploring complex networks. *Nature*, 410(6825):268–276, 2001.

[55] K. Wendy Tang and S. A. Padubidri. Diagonal and toroidal mesh networks. *IEEE Trans. Comput.*, 43(7):815–826, 1994. ISSN 0018-9340.

[56] V. Batagelj and U. Brandes. Efficient generation of large random networks. *Phys. Rev. E*, 71(3):36113, March 2005.

[57] B. W. Arden and K. Wendy Tang. Representations and routing for Cayley graphs [computer networks]. *IEEE Trans. Commun.*, 39(11):1533–1537, November 1991.

[58] K. Wendy Tang and B. W. Arden. Representations of Borel Cayley graphs. *SIAM Journal on Discrete Mathematics*, 6(4):655–676, 1993. ISSN 0895-4801.

[59] Joseph J. Rotman. *An Introduction to the theory of groups.* Springer, 4th edition, 1999.

[60] E. Angerson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammarling, J. Demmel, C. Bischof, and D. Sorensen. LAPACK: A portable linear algebra library for high-performance computers. In *Proceedings of ACM/IEEE conference on Supercomputing (Supercomputing'90)*, pages 2–11. IEEE Computer Society, 1990. ISBN O-69791-412-O.

[61] Richard Beigel, Grigorii Margulis, and Daniel A. Spielman. Fault diagnosis in a small constant number of parallel testing rounds. In *Proceedings of the Fifth annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'93)*, pages 21–29, New York, NY, USA, 1993. ACM. ISBN 0-89791-599-2.

[62] Christian Bettstetter. On the connectivity of ad hoc networks. *The Computer Journal*, 47(4):432–447, 2004.

[63] Sheldon M. Ross. *Introduction to probability models.* Academic Press, 9th edition, 2006.

[64] D.V. Chudnovsky, G.V. Chudnovsky, and M.M. Denneau. Regular graphs with small diameter as models for interconnection networks. In *Proceedings Third International Conference on Supercomputing (ICS'88)*, pages 232–239, 1988.

[65] Mirela Marta and Mihaela Cardei. Using sink mobility to increase wireless sensor networks lifetime. In *Proceedings of the International Symposium on a World of Wireless, Mobile and Multimedia Networks (WowMom'08)*,

pages 1–10, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-1-4244-2099-5.

[66] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[67] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959. ISSN 0029-599X.

[68] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8:261–277, 1988. ISSN 0209-9683.

[69] NG De Bruijn. A combinatorial problem. *Kibern. Sb., Nov. Ser*, 6:33–40, 1969.

[70] Ricardo Baeza-Yates and Raghu Ramakrishnan. Data challenges at yahoo! In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, EDBT '08, pages 652–655, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-926-5.