INFORMATION PROPAGATION ALGORITHMS FOR CONSENSUS FORMATION IN DECENTRALIZED MULTI-AGENT SYSTEMS

by

CHRISTOPHER D. HOLLANDER
B.A. Mathematics, University of South Florida, 2004
M.S. Modeling and Simulation, University of Central Florida, 2007

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2015

Major Professor: Annie S. Wu

ABSTRACT

Consensus occurs within a multi-agent system when every agent is in agreement about the value of some particular state. For example, the color of an LED, the position or magnitude of a vector, a rendezvous location, the most recent state of data within a database, or the identity of a leader are all states that agents might need to agree on in order to execute their tasking.

The task of the decentralized consensus problem for multi-agent systems is to design an algorithm that enables agents to communicate and exchange information such that, in finite time, agents are able to form a consensus without the use of a centralized control mechanism.

The primary goal of this research is to introduce and provide supporting evidence for Stochastic Local Observation/Gossip (SLOG) algorithms as a new class of solutions to the decentralized consensus problem for multi-agent systems that lack a centralized controller, with the additional constraints that agents act asynchronously, information is discrete, and all consensus options are equally preferable to all agents. Examples of where these constraints might apply include the spread of social norms and conventions in artificial populations, rendezvous among a set of specific locations, and task assignment.

This goal is achieved through a combination of theory and experimentation. Information propagation process and an information propagation algorithm are derived by unifying the general structure of multiple existing solutions to the decentralized consensus problem. They are then used to define two classes of algorithms that spread information across a network and solve the decentralized consensus problem: buffered gossip algorithms and local observation algorithms. Buffered gossip algorithms generalize the behavior of many push-based solutions to the decentralized consensus problem. Local observation algorithms generalize the behavior of many pull-based solutions to the decentralized consensus problem. In the language of object oriented design, buffered gossip al-

gorithms and local observation algorithms are abstract classes; information propagation processes are interfaces. SLOG algorithms combine the transmission mechanisms of buffered gossip algorithms and local observation algorithms into a single "hybrid" algorithm that is able to push and pull information within the local neighborhood. A common mathematical framework is constructed and used to determine the conditions under which each of these algorithms are guaranteed to produce a consensus, and thus solve the decentralized consensus problem. Finally, a series of simulation experiments are conducted to study the performance of SLOG algorithms. These experiments compare the average speed of consensus formation between buffered gossip algorithms, local observation algorithms, and SLOG algorithms over four distinct network topologies.

Beyond the introduction of the SLOG algorithm, this research also contributes to the existing literature on the decentralized consensus problem by: specifying a theoretical framework that can be used to explore the consensus behavior of push-based and pull-based information propagation algorithms; using this framework to define buffered gossip algorithms and local observation algorithms as generalizations for existing solutions to the decentralized consensus problem; highlighting the similarities between consensus algorithms within control theory and opinion dynamics within computational sociology, and showing how these research areas can be successfully combined to create new and powerful algorithms; and providing an empirical comparison between multiple information propagation algorithms.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

In a multi-agent system, consensus occurs when every agent is in agreement about the value of some particular state. Examples of such states are the color of an LED, the position or magnitude of a vector, a rendezvous location, the most recent state of data within a database, or the identity of a leader.

In the decentralized consensus problem for multi-agent systems, one is tasked with designing a method that enables agents to communicate and exchange information such that, in finite time, every agent possesses the same information without using a centralized control mechanism [1–4]. Decentralization is important because it offers a degree of simplicity, scalability, and robustness to error that cannot be easily obtained with a centralized approach [5,6]. The decentralized consensus problem is one of the most important problems in multi-agent systems, because many system-level actions require that one or more agents to coordinate and cooperate. This coordination and cooperation can only occur if all agents involved possess the same information [1,3,4].

In this dissertation, I propose the Stochastic Local Observation/Gossip (SLOG) algorithm as a new solution the decentralized consensus problem for multi-agent systems when the consensus options are discrete, agents act asynchronously, and all options for consensus are equally preferable to all agents. As a note on terminology, I refer to the information agreed upon by all agents as the *consensus state*, and use the term *consensus option* to describe any piece of information that is exchanged by agents and may therefore become the consensus state.

Rendezvous is one specific application area of the consensus problem that can benefit from the SLOG algorithm. In the rendezvous problem, a population of agents tries to agree on a specific location that every agent should meet at. This location can be either continuous or discrete. Examples of continuous locations might be a GPS location or a a set of map coordinates. A discrete

location could be a waypoint or a specific landmark. The typical solution for the rendezvous problem in the continuous domain is to have each agent use a gossip algorithm that averages incoming information with existing information [2, 3, 7–9]. In the discrete domain, the rendezvous problem has been solved by using an approach that is common in opinion dynamics: each agent surveys its neighbors and uses a heuristic to update its state based on those observations [9, 10]. Hollander and Wu [11] have also shown that gossip algorithms can be solve the rendezvous problem in the discrete domain. The SLOG algorithm can also be applied to the rendezvous problem because it too is a solution to the decentralized consensus problem in the discrete domain. Furthermore, because the SLOG algorithm combines aspects of opinion dynamics and gossip algorithms, it should be able to solve the decentralized consensus problem faster than a gossip algorithm, and without the risk of failure that exists in some opinion dynamics models.

Prelude to the Stochastic Local Observation/Gossip (SLOG) Algorithm

The need for the Stochastic Local Observation/Gossip (SLOG) algorithm can be traced back to a review of the literature on gossip algorithms that I conducted while researching normative multi-agent systems [12, 13].

In the existing literature on gossip algorithms, both within the continuous and discrete domains, there is an assumption that agents process information as soon as that information is received - they do not accumulate data by buffering incoming transmissions or storing past information in memory [14–21]. As a result of this assumption, it is unclear what should happen if an agent that is executing a gossip algorithm receives two or more incoming transmissions at the same time. The existing literature has no answer to this question, but it seems to suggest that, in practice, the reception of multiple simultaneous transmissions is typically treated as noise and all incoming simultaneous transmissions are ignored. This approach enables the formation of consensus, but it

prevents an agent from reacting based on the aggregation of the information contained within those transmissions, and thereby imposed limits on an agent's level of intelligence.

If agents are able to accumulate and respond to multiple pieces of information, then perhaps those responses could be engineered to increase the speed of consensus. It is well known within the field of AI that heuristics can often speed up the search for a solution, and that the use of such heuristics requires the accumulation of data. If gossip algorithms are only using a single data point for their decision making, then surely there is room for improvement. I developed the buffered gossip algorithm in response to this idea and applied it to solving the consensus problem in the case where the consensus options are discrete and agents act asynchronously. Simulation experiments comparing the performance of the buffered gossip algorithm against the uniform gossip algorithm [11, 22] find that using a buffer can substantially increase in the speed of consensus.

The buffered gossip algorithm, which allows agents to accumulate incoming transmissions, is functionally similar to some of the opinion dynamics models that are studied in computational sociology [7–10, 23–25]. In a buffered gossip algorithm, each agent transmits to only a single neighbor, but it is possible for an agent to to be the transmission target for multiple neighbors - i.e. an agent can receive multiple transmissions. In many opinion dynamics models, each agent queries information from every neighbor - i.e. every agent receives multiple transmissions. The similarity between the buffered gossip algorithm and opinion dynamics models begs the question: "how does the buffered gossip algorithm compare to similar opinion dynamics models on the decentralized consensus problem?"

Gossip algorithms and opinion dynamics models originate from two very different domains, and the mathematical tools used to study them are often quite different. To compare these algorithms "apples to apples", I develop the idea of an information propagation process to abstract the functionality of gossip algorithms and opinion dynamics models. I adapt the widely-used terms of

push and pull to refer to specific types of information propagation processes: push-based processes behave similar to gossip algorithms; pull-based processes behave similar to opinion dynamics models. Because an information propagation process is abstract, I define an information propagation algorithm to be a concrete representation of an information propagation process: the buffered gossip algorithm is a push-based information propagation algorithm; many opinion dynamics models are pull-based information propagation algorithms. Using this framework, I define the local observation algorithm as a generalization of the voter model [23, 26–29] and label propagation algorithm [30]) - opinion dynamics models that are similar to the buffered gossip algorithms that I have previously studied. I then compare the performance of the buffered gossip algorithm against the local observation algorithm on the decentralized consensus problem for multi-agent systems when the consensus options are discrete and agents act asynchronously. The data from this comparison between buffered gossip and local observation is inconclusive with regards which algorithm is "best", as measured by the speed of consensus. Local observation algorithms are fast, but they do not always produce a consensus. Buffered gossip algorithms guaranteed a consensus, but they are not always as fast as a local observation algorithm.

My comparison between the buffered gossip algorithm and opinion dynamics models suggests that neither approach is universally "better" than the other. Because buffered gossip algorithms are almost identical to local observation algorithms, I am curious to know whether or not performance improvements can be obtained if one combines a buffered gossip algorithm with a local observation algorithm. In reality, the simplicity of these algorithms makes their combination trivial. The Stochastic Local Observation/Gossip (SLOG) algorithm is the implementation of this combination.

Goals and Contributions

The primary goal of this dissertation is to propose a new class of solutions for the consensus problem for multi-agent systems that lack a centralized controller, with the additional constraints that agents act asynchonously, information is discrete, and all consensus options are equally preferable to all agents (e.g. social norms [12]). The initial work described in this dissertation suggests that the proposed algorithm – the Stochastic Local Observation/Gossip (SLOG) algorithm – will have an average consensus time that is less than than either a buffered gossip algorithm or a local observation algorithm (both of which are faster than the standard uniform gossip algorithm), and that this relative performance will be universal across all multiple types of network topology.

The secondary goal of this dissertation is to formally present the research that lead to the creation of the SLOG algorithm. Toward this end, I will introduce the buffered gossip algorithm as a push-based information propagation algorithm compare the consensus behavior of the buffered gossip algorithm against the uniform gossip algorithm. I will also introduce the local observation algorithm as a pull-based information propagation algorithm, and present the results from experiments in which I compare the consensus behavior of the buffered gossip algorithm against the local observation algorithm. To the best of my knowledge, such a direct comparison between push-based and pull-based methods of information propagation algorithms has yet to be done.

With these goals in mind, this dissertation contributes to the existing literature on the decentralized consensus problem in the following ways:

1. It specifies a theoretical framework that can be used to explore the consensus behavior of push-based and pull-based information propagation algorithms. Such a framework can help practitioners evaluate potential solutions in the real world.

2. It introduces buffered push-based information propagation algorithms, which allow agents

to buffer incoming transmissions, and are also capable of encapsulating the behavior of traditional "unbuffered" push-based information propagation algorithms. This allows current and future push-based algorithms to be evaluated and compared against one another within a common framework.

3. It highlights the similarities between two fields of research that are often treated independent of one another (consensus algorithms within control theory and opinion dynamics within computational sociology.) and shows how they can be successfully combined to create new and powerful algorithms.

4. It compares the average consensus speed between multiple information propagation algorithms. These results provide a general notion of relative performance that can help practitioners evaluate solutions to decentralized consensus problems that are encountered in the real world – assuming that those problems meet the constraints of discreteness, asynchronicity, and solution equality as studied in this dissertation.

Methodology

I use a combination of theory and multi-agent simulation experiments to study the buffered gossip algorithm, the local observation algorithms, and the stochastic local observation/gossip algorithm.

In my research on the buffered gossip algorithm and its comparison to the uniform gossip algorithm in the context of the decentralized consensus problem, I define a mathematical model of buffered push-based information propagation algorithms and prove that they will always yield a consensus when certain conditions hold true. I use multi-agent simulation to investigate the impact of heuristic decision making in the decentralized consensus problem. I run multiple simulations in which agents attempt to reach a consensus on some value, both with and without a buffer. The

agents that do not use a buffer mimic the functionality of the uniform gossip algorithm and adopt every value they receive, as they receive it. The agents that use a buffer employ a buffered gossip algorithm and use heuristics to decide which value to adopt.

My research on the performance differences between the buffered gossip algorithm and local observation algorithm relies on multi-agent simulation to compare the behavior of agents that use push-based information propagation algorithms against the behavior of agents that use pull-based information propagation algorithms. Prior to this experimentation, however, I employ theory to show that not all pull-based information propagation algorithms are actually capable of producing a consensus.

As with my research on the buffered gossip algorithm and the local observation algorithm, my study of the SLOG algorithm also makes use of both theory and multi-agent simulation. Using the same theoretical arguments that underly buffered push-based information propagation algorithms, I will show that Stochastic Observation/Gossip algorithms are also guaranteed to produce a consensus when certain conditions hold true, and I will use multi-agent simulation to conduct a series of experiments that measure the consensus speed of these "hybrid" algorithms against the consensus speed of push and pull algorithms.

To maintain consistency and limited scope, I constrain my focus to multi-agent systems that lack a centralized controller, with the additional constraints that agents act asynchronously, information is discrete, and all consensus options are equally preferable to all agents.

Outline

Having introduced the research covered in this dissertation, I next discuss the decentralized consensus problem and its existing push-based and pull-based solutions. This background provides

context to the Stochastic Local Observation/Gossip algorithm and the work that follows. To generalize push-based and pull-based solutions to the decentralized consensus problem, I formally define information propagation algorithms as concrete implementations of an information propagation process. These definitions are followed by a series of experiments that build upon one another. First, I explore how the ability to accumulate information can increase the speed of consensus formation by comparing the buffered gossip algorithm against the uniform gossip algorithm. Second, I reflect on the similarities between the buffered gossip algorithm and the local observation algorithm and highlights their differences by comparing their performance on the decentralized consensus problem. Finally, I propose the Stochastic Local Observation/Gossip (SLOG) algorithm as a new solution to the decentralized consensus problem for multi-agent systems wherein the consensus options are discrete and agents act asynchronously. The SLOG algorithm is compared against both buffered gossip algorithms and local observation algorithms. Following these experiments, I discuss the potential for future work and present my conclusions.

CHAPTER 2: SOLUTIONS FOR THE DECENTRALIZED CONSENSUS PROBLEM

Before discussing the Stochastic Local Observation/Gossip Algorithm, it is beneficial to establish context by first presenting some of the existing solutions to the decentralized consensus problem for multi-agent systems when the consensus options are discrete and agents act asynchronously.

In the decentralized consensus problem for multi-agent systems, one is tasked with designing a method that enables agents to communicate and exchange information such that, in finite time, every agent possesses the same information without using a centralized control mechanism [1–4]. The decentralized consensus problem is one of the most important problems in multi-agent systems, because many system-level actions require that one or more agents to coordinate and cooperate. This coordination and cooperation can only occur if all agents involved possess the same information [1, 3, 4].

Solutions to the decentralized consensus problem can be broken down into six categories based on the type of information (continuous valued or discrete valued) and the communication scheme between agents (push-based, pull-based, or a combination of both). In multi-agent systems, *push-based* communication occurs when one agent transmits information directly to one or more neighboring agents. *Pull-based* communication occurs when one agent observes or requests (and receives) information from one or more neighboring agent. Communication that is both push-based and pull-based occurs when an agent both gathers information from and transmits information to its neighbors.

The existing literature covers different aspects within four of these solution categories: consensus formation in the continuous domain with pull-based communication between agents [7–9]; con-

sensus formation in the discrete domain with pull-based communication between agents [9, 10]; consensus formation in the continuous domain with push-based communication between agents [2, 3, 18, 19]; and, consensus formation in the discrete domain with push-based communication between agents [5, 11, 19–21, 26, 31, 32]. We have found no evidence of any significant study on solutions to the consensus problem that use push-pull communication between agents in either the continuous or discrete domain.

Given the breadth of study on consensus formation in the continuous domain, and my primary goal of establishing context for the SLOG algorithm, I focus this section on existing methods consensus formation in the discrete domain – where the consensus options are discrete and cannot be averaged together or otherwise recombined. First, I discuss the prominent push-based solutions to the decentralized consensus problem, and then I discuss the prominent pull-based solutions. Finally, I introduce the mathematical notation that is used in the remainder of this work.

Push-Based Solutions to the Decentralized Consensus Problem

To solve the decentralized consensus problem in the discrete domain, agents can either communicate amongst themselves, or they can elect a leader that will speak for the population. Gossip algorithms [14–19] and leader election algorithms [32–34] define how agents receive, process, and transmit information in a decentralized environment in order to implement these paradigms using push-based communication.

Solutions to the decentralized consensus problem that use gossip algorithms depend on randomness to slowly drive a system towards consensus. Agents using a gossip algorithm contain a state value, a gossip mechanism, and a gossip protocol. The *state value* stores the information being spread through the network. The *gossip mechanism* determines how the agent selects the target(s) for its

transmission. Traditionally, selection of transmission targets is done uniformly and at random, but there is no strict requirement for this practice. Three general gossip mechanisms are used in the existing literature: select a single target from the local neighborhood (randomized gossip) [4, 21, 35, 36], select a single target from the entire network (geographic gossip) [5, 6, 14–16, 37–40], or select multiple targets from the local neighborhood [21,41] (broadcast gossip). The *gossip protocol* determines what the contents of a transmission will be, and how the receiver of a transmission will use the new information to update their internal state. The specific implementation of the gossip protocol depends on the problem being solved. Gossip algorithms can be directed or undirected with respect to their transmission of information. If the gossip algorithm is directed, only the receiver changes its state value. If the gossip algorithm is undirected, then each receiver implicitly sends its original state value back to the sender and all nodes change their state value. In practice, only gossip algorithms that use the randomized or geographic gossip mechanism are undirected. Gossip algorithms that employ the broadcast gossip mechanism are always directed.

With respect to the decentralized consensus problem in the discrete domain, we are most interested in gossip protocols used for information dissemination [5,14,19–21,31,38,42] (other common protocols include those for aggregation [17,18,29] and the construction of overlay networks [19,36]). In protocols for information dissemination, the task is to design an algorithm that results in every agent having the same state value as quickly as possible. As information spreads, it can either replace the current information contained within an agent [14,19,21], or it can be stored alongside the existing information with the goal of having every agent aware of all other state values in the system [19,31]. Common applications of information spread protocols include database synchronization [14,20], balancing processor loads [5,42], and accumulating information for use by other algorithms [31].

Solutions to the decentralized consensus problem that use leader election algorithms [32–34] depend on a single entity, called the leader, to dictate a consensus value to the rest of the population.

Paxos [32,33] is the standard algorithm for leader-based consensus formation within the tech industry, finding popularity within companies such as Google, Microsoft, Amazon, and IBM. Despite this popularity, it is widely recognized that Paxos is difficult to implement, and the real-world implementations of Paxos that do exist do not reflect the theoretical simplicity of the original algorithm [34]. Many variations of Paxos have been created as a result of the various implementation challenges faced by software engineers - but they all operate according to the same general theory. Agents that implement Paxos behave according to a predefined role. They can be either a *proposer*, an *acceptor*, a *leader*, or a combination of the three. A proposer transmits potential values for consensus to the acceptors. An acceptor chooses whether or not to accept the proposed value and lets the sender of that value know if it is accepted. If a majority of acceptors accept a proposed value, then the proposer of that value may become the leader. Learners determine the consensus value by receiving information from the acceptors and identifying the value accepted by a majority of acceptors. For a full description of how Paxos works, I refer the reader to the work of Lamport [32]. One of the biggest strengths of Paxos, besides its ability to form a consensus, is that it is fault tolerant. Leaders are selected based on a majority vote, so the failure of an agent to transmit does not stop the consensus process. Leaders can also be replaced in the event that they fail. Paxos has been primarily applied to database replication in IT systems [43], but more recently it has also been proposed for consensus formation in multi-agent systems [44].

Despite their success in the literature on the consensus problem, both gossip algorithms and Paxos-derived leader election algorithms have flaws that appear in a discrete domain when agents are allowed to communicate directly with one another.

In the case of gossip algorithms, the issue of competition between values is largely neglected. The research on information dissemination when agents can only store a single value is primarily interested in the propagation speed, and it is often assumed that the systems start with only one agent containing that information; all other agents are initialized without any information. In

the decentralized consensus problem, as we study it, every agent is initialized with a different value, and those values must compete for dominance. It is unknown if the existing performance models for gossip algorithms continue to hold true in the presence of competing information. In the research that allows agents to build up information in every node, there is no certain way to know if and when every agent in a truly decentralized system has all of the information. So, there can be no guarantee that all agents will select the same value from among the information they are aware of. In large networks, this also requires that every agent maintain a large memory.

In the case of leader election algorithms, the selection of a leader must occur before consensus is possible; this raises the question, "would it be faster just to use a different consensus algorithm to choose the consensus option, instead of first picking a leader and then having that leader propagate the value through the network by using an information dissemination algorithm?" Furthermore, many leader election algorithms rely on the ability of agents to broadcast information. Standard Paxos [32, 33] and Raft [34], specifically, also require that agents be able to respond to a transmission. This requirement for bi-directional communication is a limitation that I do not assume in this study of the decentralized consensus problem.

Pull-Based Solutions to the Decentralized Consensus Problem

Models of opinion dynamics use pull-based communication to solve the decentralized consensus problem in the discrete domain via randomized information propagation or leader election. The major models of opinion dynamics that apply to the discrete domain include the threshold model, the majority rule model, the minority rule model, and the voter model.

Agents that model opinion dynamics change their state value based on the states of all of their neighbors.

In threshold models [23], agents change their state value to some value, x, only if the proportion of neighbors with a state value of x exceeds a threshold. When the threshold is set above 50% these models follow the *majority rule* [23–25]. The majority rule dictates that an agent adopts the state equal to the statistical mode of its neighbor's states. The majority rule is also studied heavily in sociophysics, with the additional assumption that the selected agent's neighbors also change their state to the majority. Opposite to the majority rule is the *minority rule* [23, 25, 45, 46], where an agent adopts the state least represented by its neighbors. The minority rule is also studied in sociophysics with the same update assumption as the majority rule.

The voter model [23, 26, 27] is an opinion dynamics model in which two agents are selected from the population and one agent adopts the state of the other. This is identical to one agent choosing another at random, and then adopting the state of the selected agent (or vice versa). Originally, the voter model used global neighborhoods, where any two agents could be chosen at random, but more recent studies on the voter model have used local neighborhoods. In voter models that use local neighborhoods, one agent is selected at random from within the entire network, and the second agent is selected from the set of neighbors that are topologically local to the first agent. There also exist nonlinear voter models that produce results not currently observed in the mainstream research on gossip algorithms [28]. These nonlinear voter models show that it is possible for systems to produce stable patterns of stripes and clusters that are not all the same value, even when their topology suggests that a consensus should be reached; i.e. the system becomes stable, but the final state does not represent a consensus. Similar results have recently been produced by imposing social behaviors on agents that transmit information via gossiping [13]. Hollander and Wu investigate what happens when social processes, such as internalization and social enforcement, are applied to directed randomized gossip algorithms. Their findings show that if agents are able to maintain two signals, with a bias that determines which signal is more likely to be transmitted, there are many circumstances in which consensus is unobtainable. When a system is unable to obtain a consensus,

majority-driven self-imposed sanctions cause it to self-organize into stable, diverse, clusters where all agents within each cluster are attempting to transmit the same signal.

The current literature on opinion dynamics suggests that both threshold models and voter models have the capability – but not always the ability – to solve the decentralized consensus problem in the discrete domain. Most of the existing research in field of opinion dynamics seeks understand how information spreads, and not necessarily to design algorithms that produce a consensus. One example that epitomizes this goal is in the Label Propagation algorithm [30, 47]. The Label propagation algorithm is an algorithm designed to detect communities within social networks. In the label propagation algorithm, an agent updates its state value to the value held by the majority of its local neighbors. In the event that there are an even number of neighbors and the is a tie in the proportional representation of a value, the agent picks a value at random from among the tied values. The Label Propagation algorithm, while simple and straight forward, is often incapable of solving the decentralized consensus problem in large populations of agents with simple communication networks (e.g. lattice or grid topologies). This inability to form a consensus seems to be caused by the use of the majority rule on networks that have groups of nodes with a high clustering coefficient linked by nodes with low clustering coefficients. Clusters of agents adopt the same value, and then reinforce that value from the inside. Because there are not enough incoming edges from other clusters, the agents will never deviate from their state value.

Notation

Within this dissertation, I use a common notation to ensure consistency among the multiple definitions, proofs, and analyses related to the SLOG algorithm. Multi-agent systems are modeled as a communication networks; nodes represent agents and edges represent the communication/interaction links between those agents. I indicate matrices and vectors with bold upper and

lowercase symbols: $\mathbf{M}$ for matrices and $\mathbf{v}$ vectors. Individual elements are indexed, non-bold, lowercase symbols: m_{ij} for matrices and v_i for vectors. The number of elements in an arbitrary set, S, is denoted $|S|$. The probability of an arbitrary event, E, is denoted $P(E)$.

CHAPTER 3: INFORMATION PROPAGATION PROCESSES AND ALGORITHMS

Information propagation processes and algorithms provide a method by which to relate gossip algorithms with models of opinion dynamics and serve as a unifying framework for the definition and discussion of buffered gossip algorithms, local observation algorithms, and Stochastic Local Observation/Gossip algorithms.

Information Propagation Processes

I define an *information propagation process* as an abstraction layer for algorithms that define how nodes receive, process, and transmit information in a decentralized environment. An information propagation process defines the general attributes and behaviors that these algorithms must have, without an emphasis on the implementation details. As a result, many algorithms can be modeled as information propagation processes. For example, the push-based and pull-based information propagation algorithms that are discussed in chapter 2 are all information propagation processes.

Formally, let $G = (V, E)$ be an arbitrary network defined by a set of nodes, V, and a set of edges, $E = \{(u, v) : u, v \in V\}$, such that node u points to node v. Let the neighbors of node u be defined as $N(u) = \{v : (u, v) \in E \wedge u \neq v\}$.

An information propagation process specifies how information is propagated over G when each node, $u \in V$, is treated as a self-contained unit with an independent timing model, state value, buffer, transfer mechanism, transfer protocol, and state update protocol. The frequency at which data is transfered along the edge (u, v) is determined by the timing model of u. The state value of node u contains one possible consensus value. The buffer of node u contains consensus values that

may be in conflict with the state value of node u. The successful transfer of data along the edge (u, v) from node u to node v is controlled by the transfer mechanism and transfer protocol of node u, and the state update protocol of node v.

Timing Models

The *timing model* of node $u \in V$ controls the rate at which node u exchanges data with neighboring nodes in accordance with its transfer mechanism and the rate at which node u updates the state value x_u in accordance with its state update protocol. A timing model can either be asynchronous or synchronous.

Under an *asynchronous timing model*, nodes activate independently of one another. For the purposes of analysis, I assume that every node in the network possesses a clock that ticks according to a Poisson process with rate $\lambda = 1$. This is equivalent to a single clock that ticks according to a Poisson process with a rate of $n = |V|$ [18]. I call the instant of time during which these n nodes act a *time step* and reserve the term *tick* to denote the advancement of a node's internal clock. One time step can be thought of as a discrete unit of time. In practice, this means that under an asynchronous timing model an average of $n\lambda$ nodes are chosen independently and uniformly, at random, to transmit their information during each time step [18]; i.e. on average, one time step consists of $n\lambda$ ticks.

Under a *synchronous timing model*, the internal clock of each node is dependent on the clock of some other node in the network. Nodes using a synchronous timing model can be configured to activate sequentially, partially in parallel, or fully in parallel with each time step.

This dissertation focuses on asynchronous timing models because they fit naturally within the context of a decentralized system in which all entities are independent of one another.

State Values

The *state value* of node $u \in V$ is defined as $x_u \in S$ where S is the set of all possible state values. For example, in the decentralized rendezvous problem, these values are rendezvous locations; in the leader election problem, these values are candidate identifiers; and, in the norm emergence problem, these values represent social norms.

Buffers

The *buffer* of node $u \in V$ stores the data that node u has received from $N(u)$ since the last tick of node u. The buffer of node u is defined as $\beta_u \subset V \times S$ such that $\beta_u = \{(v, x_v)\}$ is a set of tuples of node $v \in N(u)$ and the state value x_v as seen by node u. For example, $\beta_1 = \{(2, 11), (3, 12), (4, 11)\}$ indicates that node 1 received the state value 11 from nodes 2 and 4 and the state value 12 from node 3. For convenience, β without a subscript to denotes the set of all buffers in the network.

Transfer Mechanism

The *transfer mechanism* of node $u \in V$ is a decision rule that determines how information is exchanged between nodes. There are six basic transfer mechanisms: *uniform gossip*, where information is transmitted from node u to a single node, $v \in N(u)$; *multicast*, where information is transmitted from node u to multiple neighboring nodes, $W \subset N(u)$; *broadcast*, where information is transmitted from node u to all neighboring nodes, $N(u)$; *simple observation*, where information is transmitted from a single node, $v \in N(u)$, to node u; *partial observation*, where information is transmitted from multiple neighboring nodes, $W \subset N(u)$, to node u; and *local observation*, where information is transmitted from all neighboring nodes, $N(u)$, to node u. Uniform gossip,

multicast, and broadcast mechanisms are all *push-based*. Simple, partial, and local observation are *pull-based* mechanisms. In addition to these six mechanisms, a hybrid approach can also be used that combines two or more approaches; for example, the push-pull algorithm introduced by Karp [20] combines the push and pull mechanisms.

Transfer Protocol

The *transfer protocol* of node $u \in V$ determines what will be transmitted to the selected neighbor, $v \in N(u)$, and what that neighbor will do with the new information once it has been received. This dissertation seeks to explain the fundamental behavior of information propagation in the context of consensus formation. To this end, it focuses on transfer protocols in which the value of node u, denoted x_u, is transmitted without modification and stored as a tagged pair within the buffer of node v; i.e., if node u transmits x_u to a neighboring node, $v \in N(u)$, then x_u is stored in β_v as the tuple (u, x_u). This transmission can occur through any valid transfer mechanism.

State Update Protocol

The *state update protocol* of node $u \in V$, defined as $f : (V \times S)^n \rightarrow V \times S$, describes how x_u is derived from β_u. This derivation uses the companion functions $g : V \times S \rightarrow V$ and $h : V \times S \rightarrow S$ that extract the components of the tuple returned by f. For example, if $\beta_1 = \{(2, 11), (3, 12), (4, 11)\}$ then one possible result is that $f(\beta_1) = (4, 11)$, $g(\beta_1) = 4$, and $x_u = h(\beta_1) = 11$.

State update protocols can be either selection-based or aggregation-based. In a *section-based state update protocol*, the new state value is an element in the original state space of the system; $h(f(\beta_u)) \in S$. In an *aggregation-based state update protocol*, the new state value is not

necessarily in the original state space of the system; for example, a real number, $h(f(\beta_u)) \in \mathbb{R}$, or an object constructed from multiple elements within β_u. I focus on selection-based state update protocols because this work is on consensus formation in the discrete domain, where the consensus is chosen from a set of initial consensus options.

Information Propagation Algorithms

If an information propagation process is intended to abstract away implementation detail, then an *information propagation algorithm* defines those details with regards to how a node receives, processes, and transmits information within a decentralized environment. An information propagation algorithm is defined by assigning a specific value to the timing model, transfer mechanism, transfer protocol, and state update protocol of an information propagation process. This relationship is analogous to the relationship between a base class and a derived class within the object oriented programming paradigm. For example, information propagation algorithm can be defined as the information propagation process that uses an asynchronous timing model, a uniform gossip transfer mechanism, a transfer protocol in which the node transmits information without modification and stores information as a tagged pair of the data and the sender, and a state update protocol that selects an element from the buffer at random.

CHAPTER 4: BUFFERED AND UNIFORM GOSSIP ALGORITHMS

The uniform gossip algorithm is a standard solution to the decentralized consensus problem that makes the assumption that agents process information as soon as that information is received [14–21]. More generally, the existing literature on the decentralized consensus problem in the discrete domain does not appear to explore gossip algorithms that are able to accumulate information - such as by buffering incoming transmissions or retaining past information in memory.

Under this assumption of instantaneous processing, an agent's behavior in response to the reception of multiple simultaneous transmissions is undefined; furthermore, the inability to accumulate information for future reference limits an agent's level of intelligence. It is well known within the field of AI that heuristics can often speed up the search for a solution, and that the use of such heuristics requires the accumulation of data. If agents are able to accumulate and respond to multiple pieces of information, then those responses can be engineered to increase the speed of consensus.

I propose buffered gossip algorithms as push-based information propagation algorithms that allow agents to accumulate information between their own actions, and thereby handle the reception of multiple simultaneous transmissions and employ heuristic decision making. It is my expectation that these buffered gossip algorithms will solve the decentralized consensus problem faster than the uniform gossip algorithm. To challenge this expectation, I formally define buffered gossip algorithms and the uniform gossip algorithm as push-based information propagation algorithms. Next, I describe the experimental design and methodology that will be used to empirically compare the consensus behavior between a set of buffered gossip algorithms and the uniform gossip algorithm. Finally, I analyze and summarize the results of those experiments.

Buffered Gossip Algorithms

Buffered gossip algorithms are push-based information propagation algorithms that allow agents to accumulate information and solve the decentralized consensus problem when all consensus options are discrete and agents act asynchronously. The behavior of these buffered gossip algorithms is central to the behavior of the Stochastic Local/Observation Gossip (SLOG) algorithms, and so buffered gossip algorithms must be understood before the behavior of a SLOG algorithm can be analyzed. Towards these ends, I first define buffered gossip algorithms and describe their basic mechanics. Then, I show that a buffered gossip algorithm will successfully solve the decentralized consensus problem when a network contains a directed spanning tree, and the nodes of that network employ both an asynchronous timing model and a selection-based state update protocol.

Definition

As an information propagation algorithm, I define a *buffered gossip algorithm* by fixing the timing model, transfer mechanism, transfer protocol, and state update protocol of node u within a specific range of values. Each combination of values within this range defines a unique buffered gossip algorithm.

The timing model of a buffered gossip algorithm is either asynchronous or synchronous. In this dissertation, I limit my discussion to asynchronous timing models due to the focus on decentralized systems, and because it is often impractical to maintain the synchronization of large decentralized populations.

Buffered gossip algorithms use a uniform gossip transfer mechanism along with a transfer protocol in which node u transmits the value of x_u without modification and stores incoming information as a tagged pair within the buffer. Therefore, each node using a buffered gossip algorithm transmits

to only one neighbor at a time, and that neighbor is selected uniformly at random. Upon receiving a transmission, these nodes store the associated information in their buffer along with the identification of the sender. As a result of this transmission behavior, there is a positive probability that β_u contains more than one piece of information prior to the execution of the state update protocol (i.e. $P(|\beta_u| > 1) > 0$). This probabilistic condition is further satisfied when it is possible for a node receives multiple simultaneous transmissions from its neighboring nodes, or when a node accumulates information over a finite period of time.

Although there are many possible implementations of a state update protocol, my interest in the decentralized consensus problem in a discrete domain drives me to focus on two specific selection-based state update protocols that ensure $(g(f(\beta_u)), h(f(\beta_u))) \in \beta_u$: *proportional selection* (f_{prop}) and *maximum frequency selection* (f_{maxf}). These selection-based state update protocols are based on two well known methods of information dissemination in opinion dynamics: the "voter model" [26] and the "label propagation algorithm" [30], respectively. The implementation of each of these state update protocols produces two distinct buffered gossip algorithms. Nodes that use a buffered gossip algorithm that implements the proportional selection protocol select a single element of β_u, chosen uniformly at random and returns the associated state value. For example, if $\beta_u = \{(2,1),(3,1),(4,2)\}$ then $P(x_u = h(f_{prop}(\beta_u)) = 1) = 2/3$ and $P(x_u = h(f_{prop}(\beta_u) = 2)) = 1/3$. A buffered gossip algorithm using proportional selection is equivalent to a voter model [23, 26–29] on a network with a time-varying topology. At any given time step, t, the neighborhood of each node, u, consists only of those nodes transmitting to node u. Nodes that use a buffered gossip algorithm that implements the maximum frequency selection protocol select a single element of β_u, chosen such that $h(f_{maxf}(\beta_u))$ is the most frequently occurring state value in node u's buffer (with ties broken randomly) and $g(f_{maxf}(\beta_u))$ is a randomly chosen node associated with $f(\beta_u)$. For example, if $\beta_u = \{(2,1),(3,1),(4,2)\}$ then $P(h(f_{maxf}(\beta_u)) = 1) = 1$ with $P(g(f_{maxf}(\beta_u)) = 2) = 0$ and $P(g(f_{maxf}(\beta_u)) = 3) = 0$ with the final result that $P(x_u = 1) = 1$. A buffered gossip

algorithm using maximum frequency selection is equivalent to the Label Propagation Algorithm [29, 30] on a network with a time-varying topology. At any given time step, t, the neighborhood of each node, u, consists only of those nodes transmitting to node u.

Mechanics

Alternatively, I can define a buffered gossip algorithm through its mechanics. If $u \in V$ is a node that uses a push-based information propagation algorithm, then node u displays the following behavior: when the internal clock of node u ticks, the first thing that node u does is to update its state value according to its state update protocol; next, the updated state value is transmitted to and stored in the buffer of one or more randomly chosen neighbors; after transmission has occurred, node u clears its buffer and waits for the next tick of its internal clock. This process of updating state, transmitting from node u to neighbors $W \subseteq N(u)$, and buffer erasing is described by algorithm 1.

Algorithm 1 The Buffered Gossip Algorithm

```
1: procedure ACT(u ∈ V)
2:     x_u ← h(f(β_u))
3:     for all v ∈ W : W ⊆ N(u) do
4:         β_v ← β_v ∪ (u, x_u)
5:     end for
6:     β_u ← ∅
7: end procedure
```

If $|W| = 1$ then I define this information propagation algorithm to be a *buffered gossip algorithm*. If $1 < |W| \leq |N(u)|$ then I define this information propagation algorithm to be *buffered multicast algorithm*. Finally, if $|W| = |N(u)|$ then I define this information propagation algorithm to be *buffered broadcast algorithm*. Within the existing literature on push-based information propagation algorithms for the decentralized consensus problem, the majority of work is aimed at the case when $|W| = 1$. In this dissertation, I continue this trend by focusing on buffered gossip algorithms.

Buffered Gossip Algorithms as Solutions to the Decentralized Consensus Problem

To show that a buffered gossip algorithm can solve the decentralized consensus problem in the discrete domain, I first describe a solution framework that allows us to study how the distribution of state values within a network changes over time. I then use this framework to show that a buffered gossip algorithm will successfully solve the decentralized consensus problem when a network contains a directed spanning tree, and the nodes of that network employ both an asynchronous timing model and a selection-based state update protocol. Formally, I say that the network, G, *contains* a directed spanning tree, Ω, if Ω is a subgraph of G. Next, I will discuss the impact of noise and node failure on the ability of a buffered gossip algorithm to form a consensus. Finally, I will show that once consensus is achieved, it remains in place until externally influenced. I do not investigate alternative timing models (e.g. synchronous) or non-selection based state update protocols (e.g. averaging) within this dissertation, and I do not derive the theoretical bounds for the consensus time of a buffered gossip algorithm; however, it is possible that current work in the literature on the voter model and Label Propagation Algorithm may be useful for future research that examines this particular issue.

A Solution Framework for Buffered Gossip Algorithms

When a multi-agent system is modeled as a network of nodes, the state of the network can be represented as a vector, and linear algebra can be used to study how the distribution of state values within the network changes over time.

When every node in a network uses a buffered gossip algorithm, an *adoption matrix*, denoted $\mathbf{A}(t)$, can be used to represent the spread of information at the end of the tth time step. Let $a_{vu} = w : w \in \{0, 1\}$ denote an element in the adoption matrix. The value w determines whether

or not x_u is used by node v when determining x_v. Consequently, $\mathbf{A}(t)$ defines a weighted graph of G in which $a_{vu} = w$ indicates an edge from node u to node v with weight w.

Adoption matrices are constructed by a *network level* state update protocol of the form $F : \beta \to \mathbb{R}^{|V|\times|V|}$, where $\beta = \{\beta_1, \beta_2, \ldots, \beta_{|V|}\}$. Network level state update protocols are algorithms that simplify the analysis of an entire network by creating adoption matrices from the buffers of all nodes within a network. In the discrete domain, adoption matrices must be row stochastic so that they satisfy the conditions $\mathbf{A}(t)\mathbf{1} = \mathbf{1}$ and $a_{ij} \in \{0, 1\}$. If an adoption matrix does not satisfy these conditions, then it reflects one or more illogical state updates (e.g. an agent attempts to be in two unique places at once, or partially present in multiple locations). I can construct a row stochastic adoption matrix using the network level state update protocol $F_{network}$ (defined by algorithm 2).

Algorithm 2 The Network Level State Update Protocol, $F_{network}$

```
function F_network( β )
    A ← 0
    for all v ∈ V do
        u ← g(f(β_v))
        a_vu ← 1
    end for
    if Σ_u a_vu = 0 then
        a_vv = 1
    end if
    return A
end function
```

In $F_{network}$, f is the selection-based state update protocol of an individual node (e.g. proportional selection or maximum frequency selection) and g is the node selection companion function. Once an adoption matrix has been constructed, the rows indicate which state values node v used to determine x_v at the end of the tth time step and the columns indicate which nodes received x_u at the end of the tth time step. For example, if f is proportional selection, then for each node $v \in V$, $a_{vu} = 1$ where node u is chosen uniformly from $g(\beta_v)$. Similarly, if f is maximum frequency selection, then for each node $v \in V$, $a_{vu} = 1$ where node u is chosen such that it is associated with

the most frequently occurring state value present in β_v (i.e. mode $(h(\beta_v))$).

Using these adoption matrices, I can study how the distribution of state values changes over time within a network of nodes that all use a buffered gossip algorithm. I can model these changes as the evolution of the linear system

$$\mathbf{x}(t+1) = \mathbf{A}(t)\mathbf{x}(t) \tag{4.1}$$

where $\mathbf{x}(t)$ is the state vector of the nodes at the end of the tth time step. Under these dynamics, the decentralized consensus problem is solved when $\mathbf{x}(t+1) = \mathbf{x}(t) = \kappa\mathbf{1}$, where κ is the *consensus state* of the system.

Convergence to a Consensus State

The first step in showing that a buffered gossip algorithm is capable of solving the decentralized consensus problem is to identify the conditions under which a consensus will form within a network of nodes using the algorithm. When using a push-based information propagation algorithm, this can occur as the result of an *information cascade*: when the state value of one node is propagated to every other node in the network. A *consensus sequence* specifies an ordered sequence of adoptions that cause an information cascade.

Definition 1 *A* consensus sequence *is a finite set* $A_\kappa = \{\mathbf{A}(t_1), \mathbf{A}(t_2)...\mathbf{A}(t_n)\}$ *with* $0 \leq t_1 < t_2 < \cdots < t_n < \infty$ *such that* $\kappa\mathbf{1} = \mathbf{A}(t_n)\cdots\mathbf{A}(t_2)\mathbf{A}(t_1)\mathbf{x}(t_1)$. *A consensus sequence specifies an ordered sequence of adoptions that propagate a single value to every node in the network.*

For any specific network, there may be multiple consensus sequences. Semantically, each matrix in a consensus sequence can be associated with an adjacency matrix that represents a path in G.

These paths denote the flow of information between nodes at the end of the associated time step.

I will now show that if a finite network, G, contains a directed spanning tree and if the nodes in G use a buffered gossip algorithm with a selection-based state update protocol and an asynchronous timing model, then at least one consensus sequence exists (lemma 1) and state information will eventually be transmitted according to that sequence (lemma 2).

Lemma 1 *If a finite network, G, has a directed spanning tree and if the nodes in G use a buffered gossip algorithm with a selection-based state update protocol and an asynchronous timing model, then a consensus sequence exists.*

Proof 1 *The proof of lemma 1 is similar to a breadth first search.*

Let $G = (V, E)$ be a finite network, let Ω be a directed spanning tree of G with root $\omega \in V$, and let d be the number of nodes that will act during time step t.

Because d is Poisson distributed when an asynchronous timing model is used, $P(d = 1)^n > 0$ if $n > 0$ is finite (i.e. there is a positive probability that only one node will be active n times in a row). Because Ω is a directed spanning tree of G with root ω, G is connected and there is at least one path from ω to every other node in the network. Because nodes act independently and $P(d = 1) > 0$, $P(\omega$ acts alone$) > 0$ at some time step $t \geq 0$ (i.e. it is possible for ω to be the only node to act during an arbitrary time step). Likewise, if the k children of ω are enumerated as $w_1 \ldots w_k$, then $P(\omega$ acts alone$)^k > 0$ for some $t \geq 0$. Because all nodes use a buffered gossip algorithm, each time ω acts it will transmit to one and only one neighbor. Because neighbors are selected uniformly at random, if $k > 1$ there is a positive probability that the selected neighbor will not have already received x_ω in the previous k time steps. Thus, after k time steps, $\beta_{w_i} = \{(\omega, x_\omega)\}$ for the ith child of ω (i.e. it is possible for ω to sequentially transmit its state value to each child, one after the other).

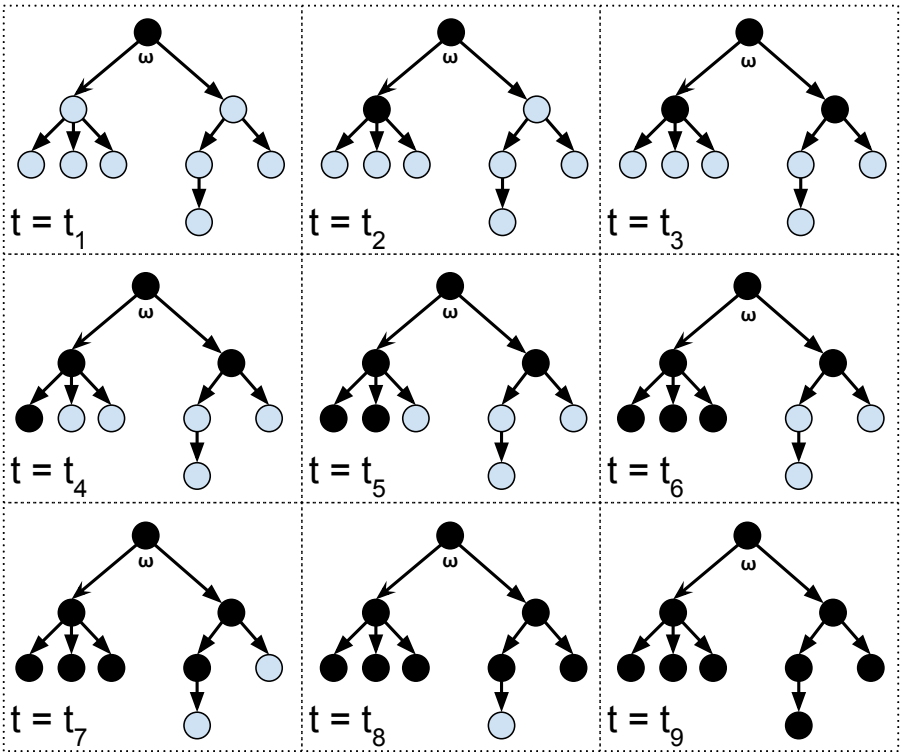

Figure 4.1: Transmission along the nodes of a directed spanning tree, Ω, with root node, ω.

Similarly, there is a positive probability that after ω has executed k transmissions, each child of ω, w_i, will act k' time steps in a row and pass along x_ω to their k' children, because node w_i will adopt x_ω as their own state since all nodes use a selection-based state update protocol and $\beta_{w_i} = \{(\omega, x_\omega)\}$. This process will continue recursively until x_ω has been adopted by every node in the network, one level at a time, moving from root to leaf.

There are n adoption matrices corresponding to all of these single-node actions. n is finite because G is finite. Thus, the finite set of these matrices form one possible consensus sequence. ■

Figure 4 visualizes the transmission process along Ω that is described in the proof of lemma 1. Given a network, G, that has a directed spanning tree, Ω, the root node, ω, starts out in state black

and proceeds to transmit that information to its children over the next two ticks. Those child then pass along the black information to their children over the next five ticks. Finally, consensus is achieved when the last node adopts the black information during the ninth tick.

Lemma 2 *If a consensus sequence exists, then it is guaranteed to be observed in asymptotic time.*

Proof 2 *Let E_i be the event "A consensus sequence is observed during the time period $\Delta t = (i, i + |A_\kappa|]$." E_i is independent from E_{i+1} because each node acts independently of one another and of past histories. Furthermore, $P(E_i) > 0$ for all $i \geq 0$ because lemma 1 establishes the existence of A_κ. Thus, $\sum_{t=0}^{\infty} P(E_t) = \infty$. Hence, by the second Borel-Cantelli Lemma, the number of observations of E_i will approach infinity as $t \to \infty$ and so the probability of observing a consensus sequence in asymptotic time is* 1. ■

Combining lemma 1 and lemma 2, I can now state the criteria for consensus under a buffered gossip algorithm.

Theorem 3 *If a finite network, G, contains a directed spanning tree and if the nodes in G use a buffered gossip algorithm with a selection-based state update protocol and an asynchronous timing model, then consensus will be obtained in asymptotic time.*

Proof 3 *By direct application of lemma 1 and lemma 2.* ■

It should be noted that, in practice, a consensus sequence does not always reflect a tree. Initial configurations and simultaneous action can lead to actual consensus sequences that are much shorter than one might expect based on the naive sequences constructed in lemma 1.

Robustness to Noise and Node Failure

The second step in showing that a buffered gossip algorithm is capable of solving the decentralized rendezvous problem is to show that it is robust to noise and node failure. Noise occurs when, for whatever reason, incorrect information is either transmitted or received. Node failure occurs when a node stops transmitting.

Theorem 3 implies that noise will not prevent consensus, but it may interfere with the formation of a consensus sequence and thus reduce the speed at which consensus occurs. Because the information transmitted between nodes may not be accurate in the presence of noise, partially formed consensus sequences may be broken. Because noise is random, however, there is a positive probability that a consensus sequence is able to form without disruption, and so lemma 1 and lemma 2 continue to hold. One interesting consequence of the buffered gossip algorithm's robustness to noise is that even though consensus will be obtained, it is possible that the final consensus state is an error value. Typically, this is undesirable behavior - but it could be leveraged by intelligent social agents as the basis of creativity, exploration, and innovation.

Theorem 3 also implies that node failure will only prevent consensus when two conditions hold: 1) the node(s) that fail are cut points within every possible directed spanning tree of G; i.e. their removal results in the inability to construct a directed spanning tree in G; and 2) the node(s) that fail never reactivate. If both of these conditions do not hold, then node failure will only delay the formation of a consensus by the same argument given on the impact of noise.

These conclusions align with the existing knowledge that robustness to noise and node failure is one of the major strengths of a gossip-based approach to consensus formation [5, 6, 20].

Stability of the Consensus State

The final step in showing that a buffered gossip algorithm is capable of solving the decentralized consensus problem is to show that once a consensus has been obtained, the consensus will be maintained until new information becomes available. Theorem 3 establishes that buffered gossip algorithms are capable of solving the decentralized consensus problem by achieving consensus in the context of locational information, but it does not ensure that the system will maintain that consensus once it has been obtained.

Lemma 4 ensures that if the system achieves consensus, it will remain in consensus until acted upon by external forces.

Lemma 4 *If a finite network, G, contains a directed spanning tree and if the nodes in G use a buffered gossip algorithm with a selection-based state update protocol and an asynchronous timing model, then $\mathbf{x}_c = \kappa\mathbf{1}$ is a fixed point of $\mathbf{x}(t+1) = \mathbf{A}(t)\mathbf{x}(t)$.*

Proof 4 *By construction, $\mathbf{A}(t)$ is row stochastic, so $\mathbf{A}(t)\mathbf{1} = \mathbf{1}$. Thus, $\mathbf{1}$ is an eigenvector of $\mathbf{A}(t)$ with an eigenvalue of $\lambda = 1$. Because scalar multiples of eigenvectors are also eigenvectors, $\mathbf{x}_c = \kappa\mathbf{1}$ is an eigenvector of $\mathbf{A}(t)$ with an eigenvalue $\lambda = 1$. So $\mathbf{A}(t)\kappa = \kappa$, and thus the consensus state, κ, is a fixed point of $\mathbf{x}(t+1) = \mathbf{A}(t)\mathbf{x}(t)$.* ■

Thus, a buffered gossip algorithm with a selection-based state update protocol and an asynchronous timing model is a solution to the decentralized consensus problem in the discrete domain if the finite network, G, contains a directed spanning tree.

Uniform Gossip Algorithms

The uniform gossip algorithm [5, 14, 15] can be modeled as a push-based information propagation algorithm in which an agent processes information as soon as that information is received. Using a state update protocol that I call tail selection, it is possible to enable the instantaneous processing of information and still account for the presence of a buffer. An additional benefit of using tail selection is that the uniform gossip algorithm can be treated as a buffered gossip algorithm. This relationship between the uniform gossip algorithm and buffered gossip algorithms is important because it allows us to directly compare the algorithms against one another and explore the impact that a buffer can have on the speed at which the decentralized consensus problem is solved. Before this comparison can be made, however, I will first define the uniform gossip algorithm as a specific type of buffered gossip algorithm. I will then show that the uniform gossip algorithm will successfully solve the decentralized consensus problem under the same conditions as the buffered gossip algorithm.

Definition

In the original description of the uniform gossip algorithm [20], nodes transmit their state value to a neighbor that has been selected according to a uniform distribution, and that neighbor then immediately updates its own state value to reflect the newly received information. As a result of this process, the state value of each node at the end of a time step reflects the last transmission that it received.

The behavior of the uniform gossip algorithm can be modeled as a push-based information propagation algorithm by using a buffered gossip algorithm in which the buffer is ordered by transmission sequence and *tail selection* is used as the state update protocol. Tail selection, denoted f_{tail},

is a selection-based state update protocol that selects the last element in the buffer. For example, if $\beta_u = \{(2,1),(3,1),(4,2)\}$ then $P(x_u = h(f_{tail}(\beta_u) = 2)) = 1$. Nodes using tail selection are equivalent to nodes that lack a buffer for long-term storage and overwrite their state value in response to every transmission. As such, the uniform gossip algorithm is only affected by the randomness of the incoming transmissions. The uniform gossip algorithm uses the same gossip mechanism and gossip protocol as the buffered gossip algorithm. Behaviorally, the use of the tail selection state update protocol causes the uniform gossip algorithm to behave like a voter model on a network with a time-varying topology. At any given time step, t, the neighborhood of each node, u, consists only of those nodes transmitting to node u.

Because the uniform gossip algorithm is a buffered gossip algorithm, its mechanical definition is identical to the mechanical definition of a buffered gossip algorithm, as defined in algorithm 1. As with the buffered gossip algorithm, it is possible for $|W|$ to vary; however, the exiting literature only appears to consider the case when $|W| = 1$.

Uniform Gossip Algorithms as Solutions to the Discrete Consensus Problem

Because the uniform gossip algorithm can be modeled as a buffered gossip algorithm with a selection-based state update protocol (i.e. tail selection), it is guaranteed to solve the decentralized consensus problem under the same conditions as the buffered gossip algorithm.

According to theorem 3, if a finite network, G, contains a directed spanning tree and if the nodes in G use a buffered gossip algorithm with a selection-based state update protocol and an asynchronous timing model, then consensus will be obtained in asymptotic time. The uniform gossip algorithm can be modeled as a buffered gossip algorithm with tail selection. Tail selection is a selection-based state update protocol. Therefore, if a finite network, G, contains a directed spanning tree, and if the nodes in G use the uniform gossip algorithm with an asynchronous timing model, then

consensus will be obtained in asymptotic time.

Similarly, because the uniform gossip algorithm can be modeled as a buffered gossip algorithm, it has the same robustness to noise and node failure as other buffered gossip algorithms with selection-based state update protocols, and lemma 4 provides the conditions under which a consensus formed by the uniform gossip algorithm is stable.

Therefore, the uniform gossip algorithm with an asynchronous timing model is a solution to the decentralized consensus problem in the discrete domain if the finite network, G, contains a directed spanning tree.

An Experimental Comparison between Buffered and Uniform Gossip Algorithms

Multi-agent simulation can be used to empirically compare the consensus behavior between a set of buffered gossip algorithms and the uniform gossip algorithm. The data from such a comparison can then be used to determine the impact that a buffer can have on the speed at which the decentralized consensus problem can be solved.

I have created such a simulation using Python and the NetworkX [48] and Numpy [49] libraries. In the remainder of this section, I describe the experimental design and methodology that structures my experiments on the impact of information accumulation. I then discuss my hypotheses and present the results of my simulations. Finally, I interpret those results as they related to the impact of information accumulation on the speed of consensus formation within a decentralized system of asynchronous agents.

Experimental Design and Methodology

The simulation used to explore the behavior of the uniform and buffered gossip algorithms models a multi-agent system consisting of $n = |V|$ asynchronous agents that are connected by a static communication network. Each node in the network represents an agent and an edge connects two nodes if there is a communication link between the associated agents. The state value of each node represents that node's desired consensus option and is encoded as an integer value. Each node can store up to n transmissions in its buffer, and those transmissions are stored in the order in which they are received. If a node receives multiple transmissions from the same agent before it is able to clear its buffer, only the most recent transmission is retained. To account for the diversity of many real-world networks, the communication network can be structed as either an Erdös-Renyi random network, a Barabasi-Albert scale-free network, a Newman-Watts-Strogatz small world network [50], or a lattice network.

Nodes use an asynchronous timing model, where the expected number of nodes that act in a single time step follows a Poisson distribution with $\lambda = |V|$. Because asynchronous timing models are used, it is possible that some nodes will act multiple times within a single time step. Simulation time is measured in *steps*. One step has passed when all active nodes have updated their state value and spread their information in accordance with their action algorithm. Thus, one step is equivalent to one time step. Those nodes that act within a single step do so in a uniformly random order.

The state update protocol (proportional selection, maximum frequency selection, or tail selection) and the network topology (Erdös-Renyi random, Barabasi-Albert scale-free, Newman-Watts-Strogatz small world, or lattice) are the primary independent variables. For each combination of state update protocol and network topology, I randomly construct 300 networks with the selected topological structure and then conduct 30 independent simulations of rendezvous over each network. These networks are constructed randomly, with $2 \leq |V| \leq 100$ and $2 \leq |S| \leq 5$

being chosen according to a uniform distribution. The decision to vary network and state space size was made to test solution potential over a wide range of possibilities. Additionally, Erdös-Renyi random networks use a random value in the range $[0, 1]$ for their connection probability, and are guaranteed to be connected; Barabase-Albert scale-free networks and Newman-Watts-Strogatz small world networks are randomly parameterized based on the number of nodes in the network; and lattice networks are guaranteed to be square and do not wrap to form a torus. The parameters associated with each network topology are a requirement of the NetworkX library.

The consensus time (measured in steps) is the dependent variable under study, with the characterization that a value of $100,000$ represents a failure to achieve consensus. Nodes successfully form a consensus if the state of every node is identical within $100,000$ steps. Nodes fail to form a consensus if either periodic behavior is observed or the simulation runs in excess of a maximum time limit ($100,000$ steps). The simulation software is capable of detecting periodic behavior of up to 100 unique states. Behavior is considered to be periodic if a sequence of state distributions repeats continuously for $10,000$ consecutive steps (e.g. a sequence of 10 state distributions repeats $1,000$ times in a row).

Each simulation runs until either consensus is reached, a non-consensus stable state is observed (either fixed or periodic), or a time limit of $110,000$ steps is exceeded. This produces a total of $9,000$ data points per experimental configuration. To remove randomness as a cause for differences between experimental configurations, each configuration is initialized with same sequence of random numbers (i.e. simulation 17 of the configuration $\{$proportional, random$\}$ uses the same random seed as simulation 17 of the configuration $\{$maximum, lattice$\}$).

As with the theoretical proofs of consensus, these experiments focus on information propagation algorithms with proportional selection and maximum frequency selection state update protocols because they are similar to the voter model and the label propagation algorithm; although because

they are being used in a new context there is no guarantee that they will display the same behavior. I also make the simplifying assumption that in the event of a node receiving multiple transmissions from the same neighbor prior to a state update, only the most recent transmission is kept in the buffer. Finally, because this dissertation focuses on comparing consensus speed between different algorithms, and because noise and node failure only prevent consensus formation in very specific scenarios, I assume here that information is transmitted without error and nodes do not fail during consensus formation. This assumption simplifies the experiments by holding the noise and node failure probabilities constant at a value of 0.0.

Hypotheses

To compare the relative consensus speed between buffered and uniform gossip algorithms, I use Welch's t test [51] and ANOVA on the data generated by the multi-agent simulation to test the following hypotheses:

- Because maximum frequency selection is explicitly designed to be less random than proportional selection, I expect that the mean consensus time of a buffered gossip algorithm using maximum frequency selection (μ_{max}) is less than the the mean consensus time of a buffered gossip algorithm using proportional selection (μ_{pro}). Supporting evidence for this expectation exists if I am able to reject the null hypothesis: H_1: $\mu_{max} \geq \mu_{pro}$.

- Because maximum frequency selection is explicitly designed to be less random than the uniform gossip algorithm, I expect that the mean consensus time of a buffered gossip algorithm using maximum frequency selection (μ_{max}) is less than the the mean consensus time of the uniform gossip algorithm (μ_{uni}). Supporting evidence for this expectation exists if I am able to reject the null hypothesis: H_2: $\mu_{max} \geq \mu_{uni}$.

- Because randomness is a core component of proportional selection and the uniform gossip algorithm, I expect that the mean consensus time of a buffered gossip algorithm using proportional selection (μ_{pro}) is equal to the the mean consensus time of the uniform gossip algorithm (μ_{uni}). Supporting evidence for this expectation exists if I fail to reject the null hypothesis: H_3: $\mu_{pro} = \mu_{uni}$.

I also consider the impact of network topology on consensus speed by testing hypotheses related to four different types of networks (random, lattice, scale-free, and small world):

- Because differences in network topology have been found to affect the performance of the label propagation algorithm [30], and because the label propagation algorithm is the basis for maximum frequency selection, I expect that there will be differences between the mean consensus times of a buffered gossip algorithm using maximum frequency selection on a random network ($\mu_{max}(random)$), a scale-free network ($\mu_{max}(scale)$), a small world network($\mu_{max}(small)$), and a lattice network ($\mu_{max}(lattice)$). Supporting evidence for this expectation exists if I am able to reject the null hypothesis: H_4: $\mu_{max}(random) = \mu_{max}(lattice) = \mu_{max}(scale) = \mu_{max}(small)$.

- Because differences in network topology have been found to affect the performance of the voter model [27], and because the voter model is the basis for proportional selection, I expect that there will be differences between the mean consensus times of a buffered gossip algorithm using proportional selection on a random network ($\mu_{pro}(random)$), a scale-free network ($\mu_{pro}(scale)$), a small world network($\mu_{pro}(small)$), and a lattice network ($\mu_{pro}(lattice)$). Supporting evidence for this expectation exists if I am able to reject the null hypothesis: H_5: $\mu_{pro}(random) = \mu_{pro}(lattice) = \mu_{pro}(scale) = \mu_{pro}(small)$.

- Because differences in network topology have been found to affect at least one randomized

algorithm used in information propagation (e.g. the voter model [27]), and because randomness is a core component of the uniform gossip algorithm, I expect that there will be differences between the mean consensus times of the uniform gossip algorithm on a random network ($\mu_{max}(random)$), a scale-free network ($\mu_{max}(scale)$), a small world network($\mu_{max}(small)$), and a lattice network ($\mu_{max}(lattice)$). Supporting evidence for this expectation exists if I am able to reject the null hypothesis: H_6: $\mu_{uni}(random) = \mu_{uni}(lattice) = \mu_{uni}(scale) = \mu_{uni}(small)$.

Finally, given the established literature that illustrates the potential impacts of network topology, I expect that the relative performance of a buffered gossip algorithm using proportional selection or maximum frequency selection, and the uniform gossip algorithm, differs across network topologies. For consistency, I denote this test as H_7 and verify it by graphical analysis.

Empirical Results

This section discusses the results of my experiments on 300 randomly generated Erdös-Renyi random networks, 300 randomly generated Barabasi-Albert scale-free networks, 300 randomly generated Newman-Watts-Strogatz small world networks, and 300 randomly generated lattice networks.

Random Networks

Figure 4.2 visualizes the experimental data from 300 randomly generated Erdös-Renyi random networks using a standard box plot. The upper and lower boundaries of each box correspond to the first and third quartile of the data, with the middle line representing the median value. The upper and lower whiskers extend out to the largest and smallest value within $1.5 \times IRQ$ of the boundary. The individual points represent the outliers of the observed data. The x-axis indicates the state up-

date protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed. One can observe that a buffered gossip algorithm using maximum frequency selection has the lowest median consensus time and smallest third quartile of the three algorithms. These observations suggest that, when agents communicate over Erdös-Renyi random networks, a buffered gossip algorithm using maximum frequency selection should produce lower consensus times in comparison to a buffered gossip algorithm using proportional selection or the uniform gossip algorithm.

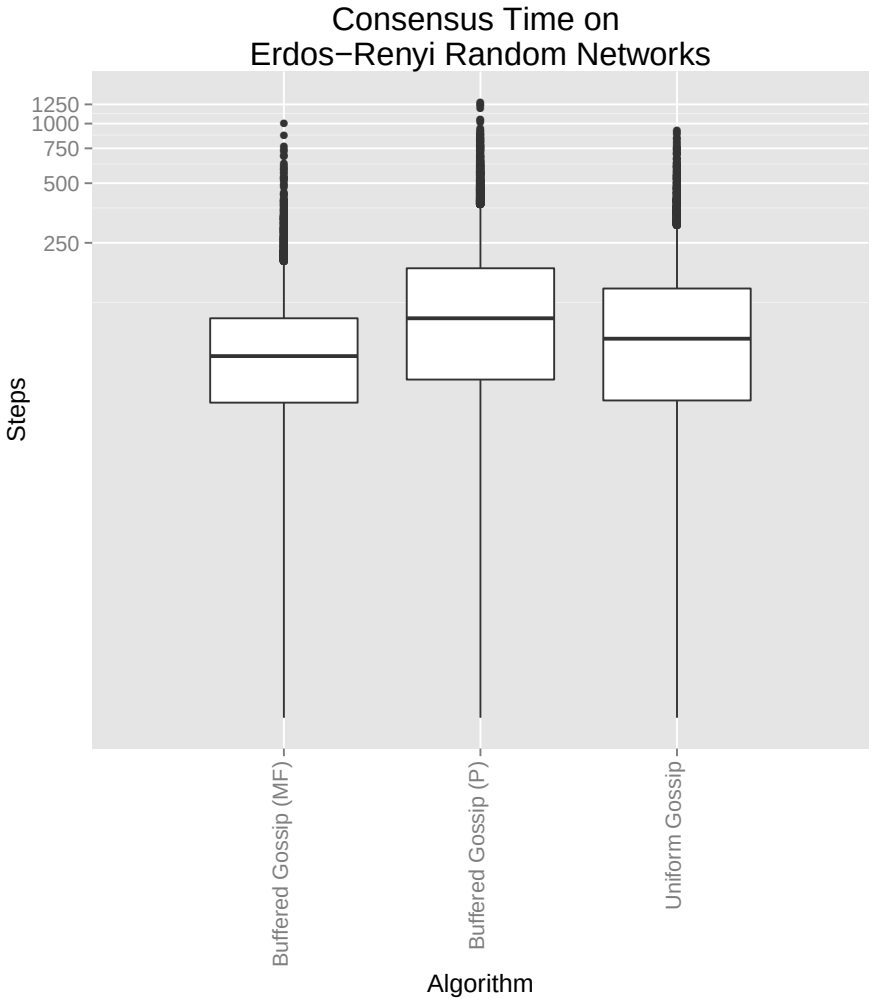

Figure 4.2: Data from the empirical comparison of the uniform and buffered gossip algorithms, visualized as box plots that show the interquartile range, median value, and outliers of the consensus time on Erdös-Renyi random networks.

Figure 4.3 visualizes the mean consensus time of the random network data along with the 95% confidence interval of each mean. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. I test hypotheses

H_1 ($\mu_{max} \geq \mu_{pro}$), H_2 ($\mu_{max} \geq \mu_{uni}$), and H_3 ($\mu_{pro} = \mu_{uni}$) in the context of Erdös-Renyi random networks using the data visualized in Figure 4.3. I reject hypotheses H_1 and H_2 ($t_{H_1}(13340.78) = -38.85$, $t_{H_2}(15499.30) = -23.00$, $p < 0.01$ for both). This suggests that there is evidence to support the claim that the mean consensus time of a buffered gossip algorithm using maximum frequency selection ($\mu_{max} = 79.12, \sigma_{max} = 64.94$) is less than the the mean consensus time of a buffered gossip algorithm using proportional selection ($\mu_{pro} = 137.91, \sigma_{pro} = 128.06$) and less than the mean consensus time of the uniform gossip algorithm ($\mu_{uni} = 107.89, \sigma_{uni} = 99.38$). I also reject H_3 ($t(16953.49) = 17.57, p < 0.01$), because there is not evidence to support the claim that the mean consensus time of a buffered gossip algorithm using proportional selection ($\mu_{pro} = 137.91, \sigma_{pro} = 128.06$) is equal to the the mean consensus time of the uniform gossip algorithm ($\mu_{uni} = 107.89, \sigma_{uni} = 99.38$). Instead, the evidence suggests that the mean consensus time of the uniform gossip algorithm is less than the mean consensus time of a buffered gossip algorithm using proportional selection. The rejection of H_3 may suggest that even though randomness is central to proportional selection and the uniform gossip algorithm, there are other factors that I have not yet examined that may influence the length of time required to form a consensus.

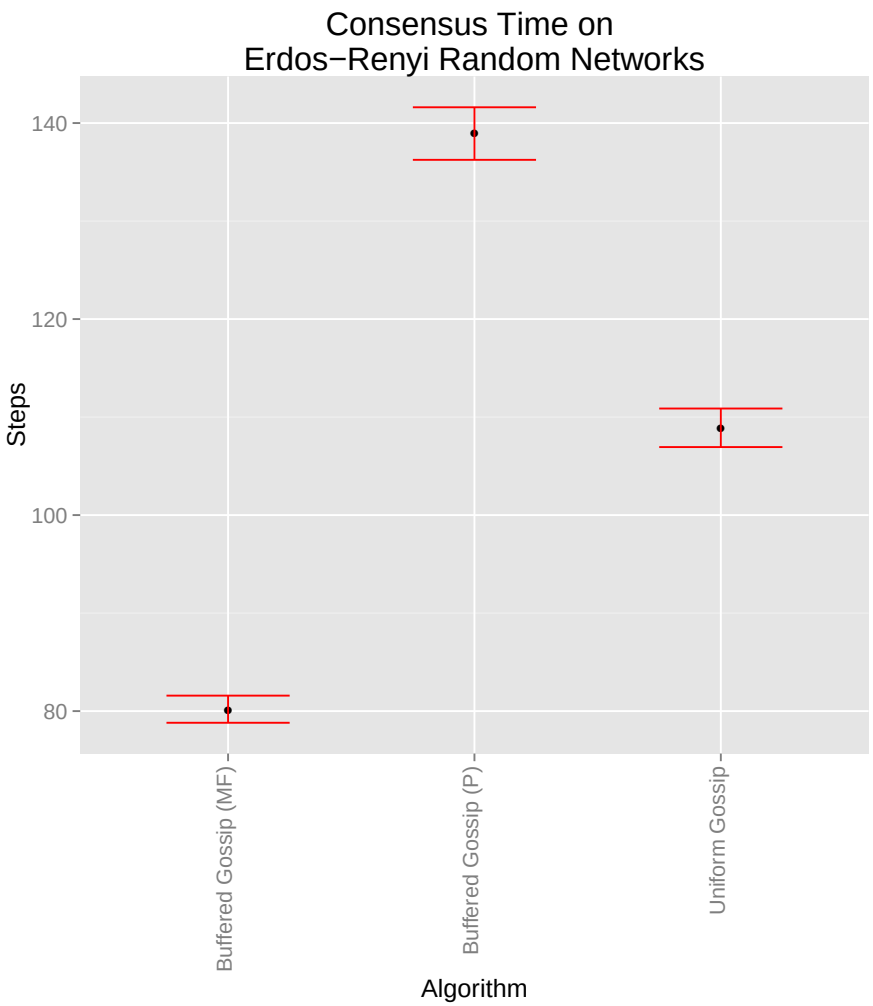

Figure 4.3: Data from the empirical comparison of the uniform and buffered gossip algorithms, visualized as errorbar plots of the 95% confidence intervals for the mean consensus time on Erdös-Renyi random networks.

Scale-Free Networks

Figure 4.4 visualizes the experimental data from 300 randomly generated Barabasi-Albert scale-free networks using a standard box plot. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed. I observe that a buffered gossip algorithm using maximum frequency selection has the lowest median consensus time and smallest third quartile. These observations suggest that, when agents communicate over Barabasi-Albert scale-free networks, a buffered gossip algorithm using maximum frequency selection should produce lower consensus times in comparison to a buffered gossip algorithm using proportional selection or the uniform gossip algorithm.

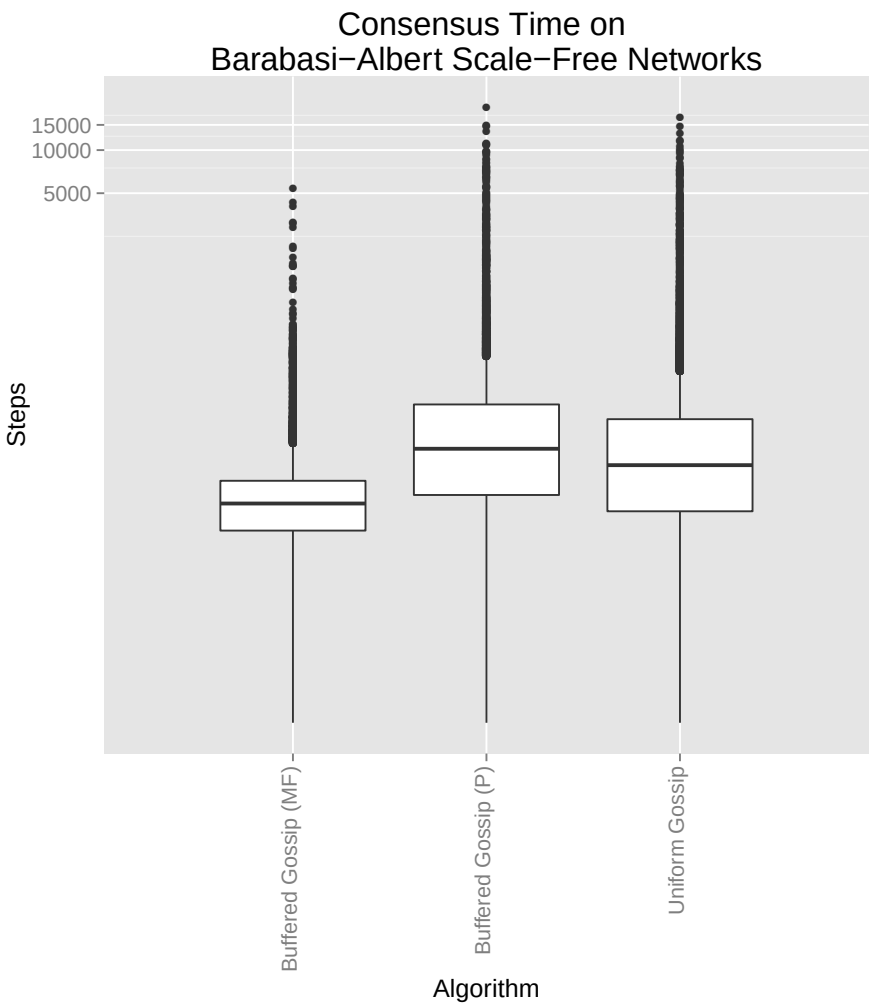

Figure 4.4: Data from the empirical comparison of the uniform and buffered gossip algorithms, visualized as box plots that show the interquartile range, median value, and outliers of the consensus time on Barabasi-Albert scale-free networks

Figure 4.5 visualizes the mean consensus time of the scale-free network data along with the 95% confidence interval of each mean. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. I test

hypotheses H_1 ($\mu_{max} \geq \mu_{pro}$), H_2 ($\mu_{max} \geq \mu_{uni}$), and H_3 ($\mu_{pro} = \mu_{uni}$) in the context of scale-free networks using the experimental data that underlies Figure 4.5. I reject hypotheses H_1 and H_2 ($t_{H_1}(9591.36) = -21.49$, $t_{H_2}(9696.97) = -19.41$, $p < 0.01$ for both). This suggests that there is evidence to support the claim that the mean consensus time of a buffered gossip algorithm using maximum frequency selection ($\mu_{max} = 47.41, \sigma_{max} = 125.68$) is less than the the mean consensus time of a buffered gossip algorithm using proportional selection ($\mu_{pro} = 206.81, \sigma_{pro} = 692.37$) and less than the mean consensus time of the uniform gossip algorithm ($\mu_{uni} = 180.40, \sigma_{uni} = 637.70$). I also reject H_3 ($t_{H_3}(17877.61) = 2.66$, $p < 0.02$), because there is not sufficient evidence to support the claim that the mean consensus time of a buffered gossip algorithm using proportional selection ($\mu_{pro} = 206.81, \sigma_{pro} = 692.37$) is equal to the the mean consensus time of the uniform gossip algorithm ($\mu_{uni} = 180.40, \sigma_{uni} = 637.70$). Instead, the evidence suggests that the mean consensus time of the uniform gossip algorithm is less than the mean consensus time of a buffered gossip algorithm using proportional selection. The rejection of H_3 may suggest that even though randomness is central to proportional selection and the uniform gossip algorithm, there are other factors that I have not yet examined that may influence the length of time required to for a consensus.

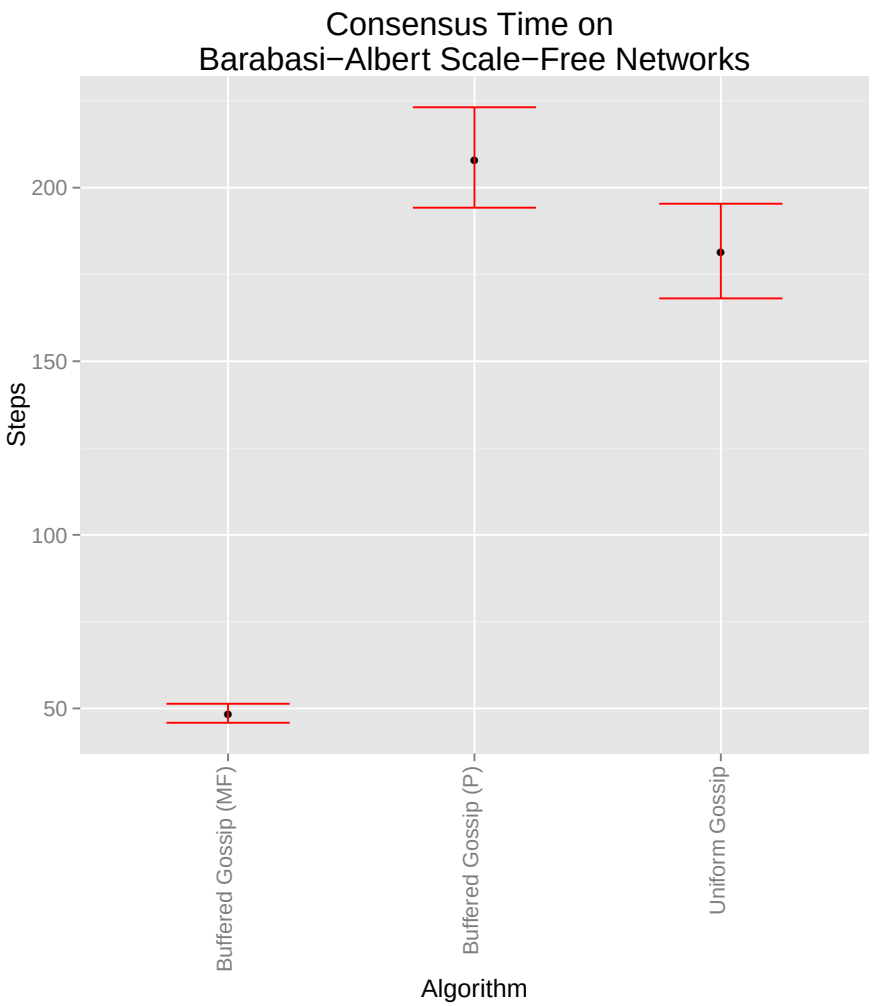

Figure 4.5: Data from the empirical comparison of the uniform and buffered gossip algorithms, visualized as errorbar plots of the 95% confidence intervals for the mean consensus time on Barabasi-Albert scale-free networks.

Small World Networks

Figure 4.6 visualizes the experimental data from 300 randomly generated Newman-Watts-Strogatz small world networks using a standard box plot. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed. I observe that a buffered gossip algorithm using maximum frequency selection has the lowest median consensus time and smallest third quartile. These observations suggest that, when agents communicate over Newman-Watts-Strogatz small world networks, a buffered gossip algorithm using maximum frequency selection should produce lower consensus times in comparison to a buffered gossip algorithm using proportional selection or the uniform gossip algorithm.

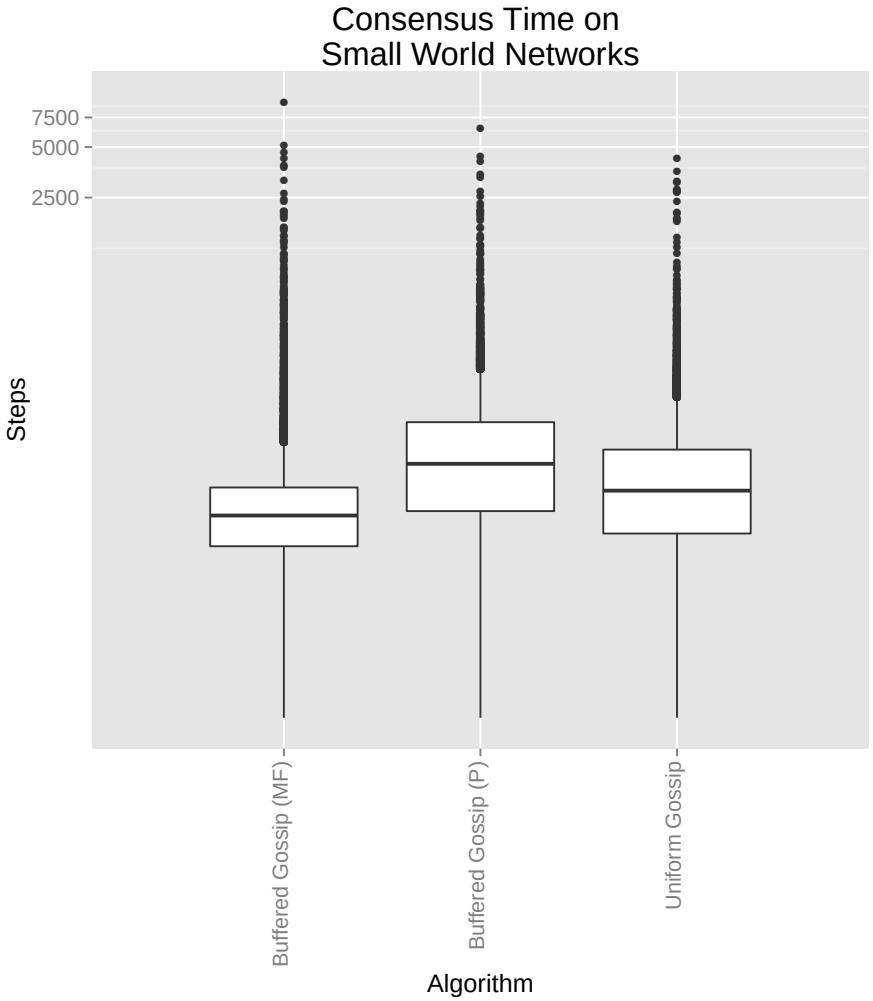

Figure 4.6: Data from the empirical comparison of the uniform and buffered gossip algorithms, visualized as box plots that show the interquartile range, median value, and outliers of the consensus time on Newmann-Watts-Strogatz small world networks

Figure 4.7 visualizes the mean consensus time of our small world network data along with the 95% confidence interval of each mean. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. I test hypotheses

H_1 ($\mu_{max} \geq \mu_{pro}$), H_2 ($\mu_{max} \geq \mu_{uni}$), and H_3 ($\mu_{pro} = \mu_{uni}$) in the context of small world networks using the experimental data that underlies Figure 4.7. I reject hypotheses H_1 and H_2 ($t_{H_1}(17809.95) = -16.83$, $t_{H_2}(16262.43) = -6.23$, $p < 0.01$ for both). This suggests that there is evidence to support the claim that the mean consensus time of a buffered gossip algorithm using maximum frequency selection ($\mu_{max} = 52.49, \sigma_{max} = 180.70$) is less than the the mean consensus time of a buffered gossip algorithm using proportional selection ($\mu_{pro} = 95.66, \sigma_{pro} = 163.00$) and less than the mean consensus time of the uniform gossip algorithm ($\mu_{uni} = 67.07, \sigma_{uni} = 128.73$). I also reject H_3 ($t_{H_3}(17080.96) = 13.06$, $p < 0.01$), because there is not sufficient evidence to support the claim that the mean consensus time of a buffered gossip algorithm using proportional selection ($\mu_{pro} = 95.66, \sigma_{pro} = 163.00$) is equal to the the mean consensus time of the uniform gossip algorithm ($\mu_{uni} = 67.07, \sigma_{uni} = 128.73$). Instead, the evidence suggests that the mean consensus time of the uniform gossip algorithm is less than the mean consensus time of a buffered gossip algorithm using proportional selection. The rejection of H_3 may suggest that even though randomness is central to proportional selection and the uniform gossip algorithm, there are other factors that I have not yet examined that may influence the length of time required to form a consensus.

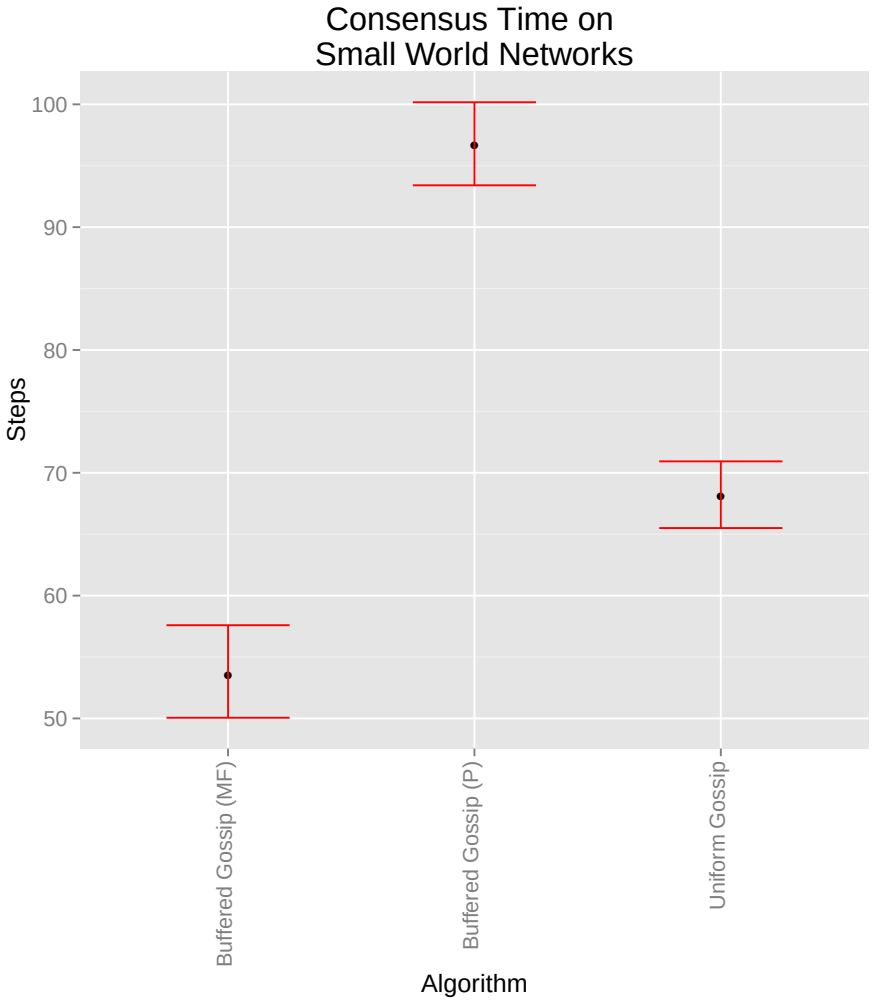

Figure 4.7: Data from the empirical comparison of the uniform and buffered gossip algorithms, visualized as errorbar plots of the 95% confidence intervals for the mean consensus time on Newmann-Watts-Strogatz small world networks.

Lattice Networks

Figure 4.8 visualizes the experimental data from 300 randomly generated lattice networks using a standard box plot. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed. I observe that the uniform gossip algorithm has the lowest median consensus time and smallest third quartile. I also observe that the total performance range (including outliers) is similar between a buffered gossip algorithm using maximum frequency selection, a buffered gossip algorithm using proportional selection and the uniform gossip algorithm; although the median value and third quartile of a buffered gossip algorithm using maximum frequency selection is less than the median value and third quartile of the proportional data. These observations suggest that when agents communicate through lattice random networks, the uniform gossip algorithm should produce lower consensus times in comparison to a buffered gossip algorithm using maximum frequency selection or proportional selection, but it would not be uncommon for all three algorithms to produce similar results.

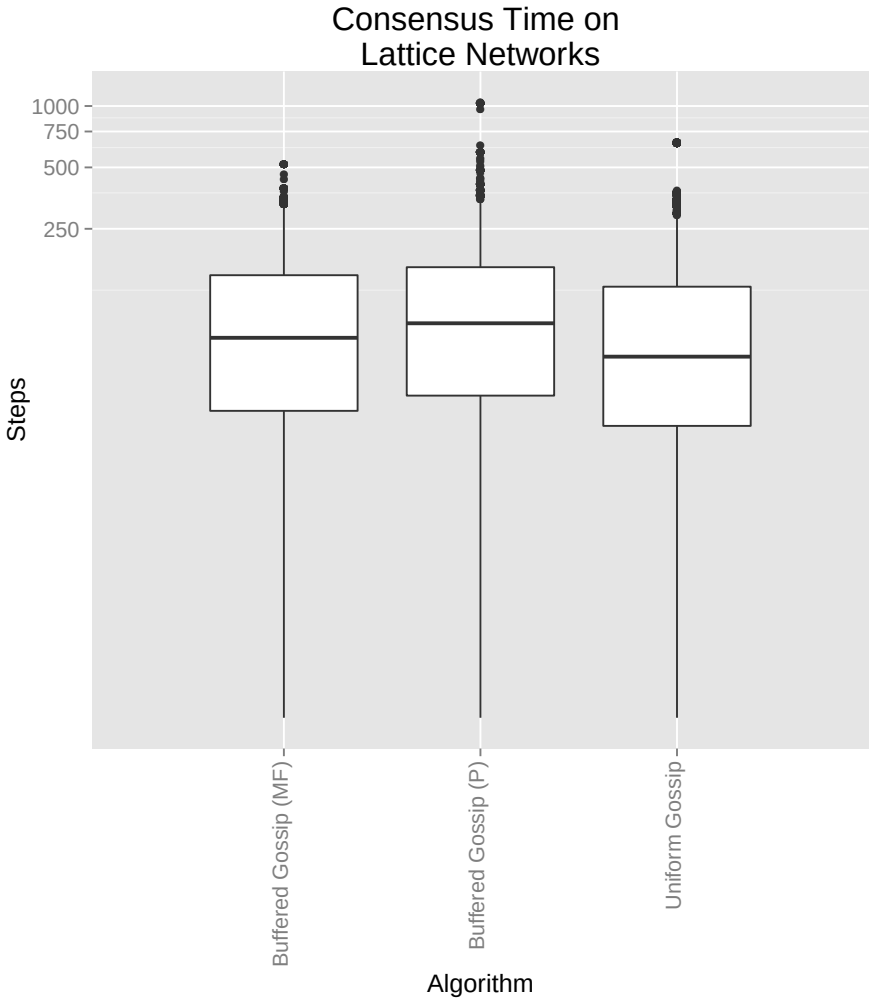

Figure 4.8: Data from the empirical comparison of the uniform and buffered gossip algorithms, visualized as box plots that show the interquartile range, median value, and outliers of the consensus time on lattice networks.

Figure 4.9 visualizes the mean consensus time of the lattice network data along with the 95% confidence interval of each mean. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. I test hypotheses

H_1 ($\mu_{max} \geq \mu_{pro}$), H_2 ($\mu_{max} \geq \mu_{uni}$), and H_3 ($\mu_{pro} = \mu_{uni}$) in the context of lattice networks using the experimental data that underlies Figure 4.9. I reject hypotheses H_1 ($t_{H_1}(16329.79) = -11.03$, $p < 0.01$). This suggests that there is evidence to support the claim that the mean consensus time of a buffered gossip algorithm using maximum frequency selection ($\mu_{max} = 101.26, \sigma_{max} = 92.48$) is less than the the mean consensus time of a buffered gossip algorithm using proportional selection ($\mu_{pro} = 119.69, \sigma_{pro} = 128.80$). I fail to reject H_2 ($t_{H_2}(17635.54) = 3.45$, $p = 1$). This suggests that there is not sufficient evidence to support the claim that the mean consensus time of a buffered gossip algorithm using maximum frequency selection ($\mu_{max} = 101.26, \sigma_{max} = 92.48$) is less than the mean rendezvous time of the uniform gossip algorithm ($\mu_{uni} = 96.12, \sigma_{uni} = 106.85$). I also reject H_3 ($t_{H_3}(17404.13) = 13.36$, $p < 0.01$), because there is also not sufficient evidence to support the claim that the mean consensus time of a buffered gossip algorithm using proportional selection ($\mu_{pro} = 119.69, \sigma_{pro} = 128.80$) is equal to the the mean consensus time of the uniform gossip algorithm ($\mu_{uni} = 96.12, \sigma_{uni} = 106.85$). Instead, as in a random network, the evidence suggests that the mean consensus time of the uniform gossip algorithm is less than the mean consensus time of a buffered gossip algorithm using proportional selection. The rejection of H_3 may suggest that even though randomness is central to proportional selection and the uniform gossip algorithm, there are other factors that I have not yet examined that may influence the length of time required to form a consensus.

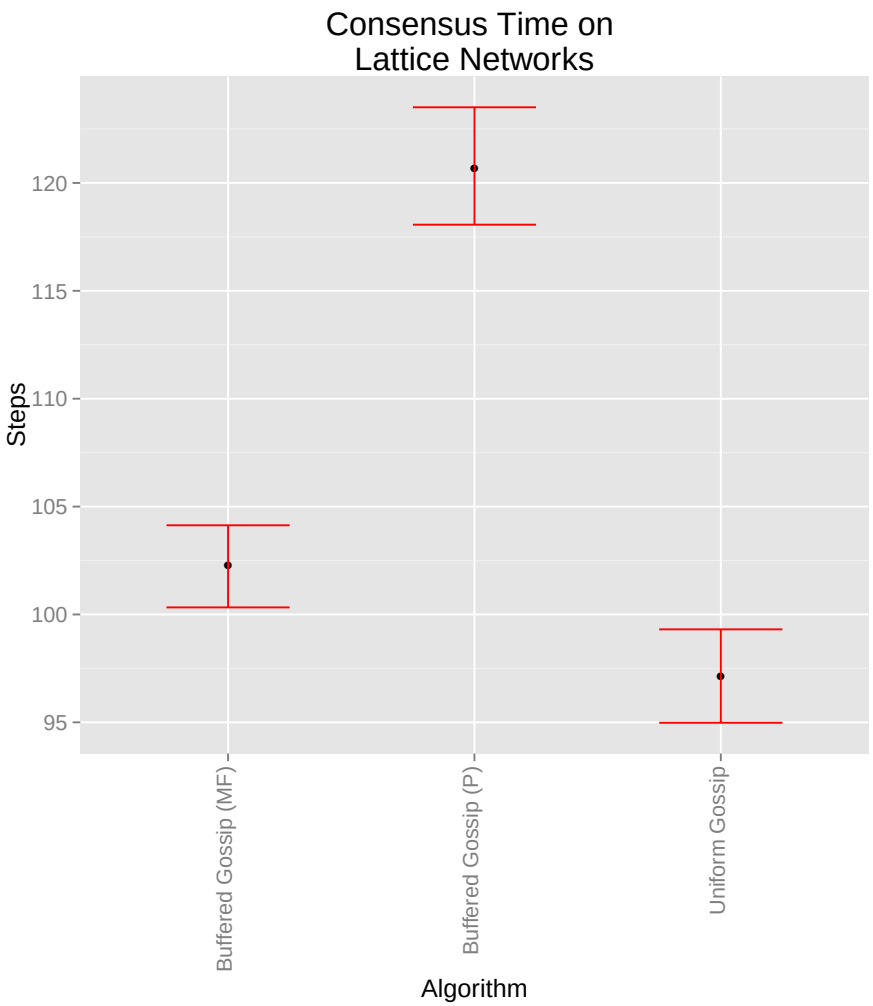

Figure 4.9: Data from the empirical comparison of the uniform and buffered gossip algorithms, visualized as errorbar plots of the 95% confidence intervals for the mean consensus time on lattice networks.

Impact of Network Topology

In regards to comparing the mean consensus time across network topology, I reject H_4, H_5, and H_6 ($p < 0.01$ for each). The evidence supports the claim that consensus time is sensitive to the topology of the agent communication network. Furthermore, one can observe evidence to support hypothesis H_7: the communication topology appears to produce a difference in the relative performance of a buffered gossip algorithm proportional selection or maximum frequency selection, and the uniform gossip algorithm.

Summary of Results

Consensus was observed in all of the experiments comparing buffered gossip algorithms against the uniform gossip algorithm. In the worst case, the maximum consensus time was 19909 steps and occurred under proportional selection over a Barabasi-Albert scale-free network with $|V| = 84$ nodes, $|E| = 83$ edges, and $|S| = 6$.

The results of these experiments suggest that while the state update protocol does exhibit influence on the consensus time, the topology of the communication network may be the most critical factor in the speed of consensus. Evidence of this behavior is found in the observed consensus times across the four network topologies tested in the experiments – the mean consensus time varies more between the type of network than the state update protocol. This finding is in line with the existing research on the voter model and Label Propagation Algorithm. Furthermore, the underlying theory of the buffered gossip algorithm also suggests that the topology is critical to the overall success of a consensus problem solution; e.g. disconnected networks will never achieve consensus.

In regards to the impact that a buffer can have on the speed of consensus, the results suggest that algorithms which store information in a buffer typically have lower mean consensus times than

those that do not. In the worst case among the results (lattice networks), a buffered gossip algorithm using maximum frequency selection is only slightly worse than the uniform gossip algorithm.

CHAPTER 5: BUFFERED GOSSIP AND LOCAL OBSERVATION ALGORITHMS

Buffered gossip algorithms, which allow agents to accumulate incoming transmissions, are functionally similar to some of the opinion dynamics models that are studied in computational sociology (e.g. the voter model or the label propagation algorithm). In a buffered gossip algorithm, each agent transmits to only a single neighbor, but it is possible for an agent to to be the transmission target for multiple neighbors - i.e. an agent can receive multiple transmissions. In many opinion dynamics models, each agent queries information from every neighbor - i.e. every agent receives multiple transmissions.

To determine whether or not the functional similarities between buffered gossip algorithms and models of opinion dynamics also lead to similarities in performance on the decentralized consensus problem, I first define the local observation algorithm as abstraction layer for models of opinion dynamics and other pull-based information propagation algorithms. Next, I describe the experimental design and methodology that is used to empirically compare the performance of local observation algorithms against the uniform gossip algorithm and buffered gossip algorithms with identical timing models, transfer protocols, and state update protocols. To the best of my knowledge, such a direct comparison between push-based and pull-based methods of information propagation algorithms has not yet been done. Finally, I analyze and summarize the results of those experiments.

Local Observation Algorithms

Local observation algorithms are pull-based information propagation algorithms can that be used to solve the decentralized consensus problem when all consensus options are discrete and agents act asynchronously. Local observation algorithms generalize models of opinion dynamics that use pull-based algorithms to obtain information by requesting it from or observing neighboring agents (as opposed to pushing their own values on to one or more specific neighbors as push-based algorithms do). The behavior of local observation algorithms is also central to the behavior of the Stochastic Local/Observation Gossip (SLOG) algorithm, and so like buffered gossip algorithms, these algorithms must be understood before defining the SLOG algorithm. Towards this end, I first define local observation algorithms and describe their basic mechanics. I then show that local observation algorithms are capable of solving the decentralized consensus problem, but there is not always a guarantee that they will.

Definition

As an information propagation algorithm, I define a *local observation algorithm* by fixing the timing model, transfer mechanism, transfer protocol, and state update protocol of node u within a specific range of values. Each combination of values within this range defines a unique local observation algorithm.

The timing model of a local observation algorithm is either asynchronous or synchronous. In this dissertation, I limit my discussion to asynchronous timing models due to the focus on decentralized systems, and because it is often impractical to maintain the synchronization of large decentralized populations.

Local observation algorithms use a local observation transfer mechanism along with a transfer

protocol in which multiple nodes, $W = N(u)$, transmit their state value to node u without modification and node u stores incoming information as a tagged pair within the buffer β_u. Therefore, each node using a local observation algorithm receives information from all neighboring nodes during a single tick. Upon receiving a transmission, these nodes store the associated information in their buffer along with the identification of the sender. As a result of this transmission behavior, the buffer of node u will contain always multiple elements (i.e. $P(|\beta_u| > 1) = 1$).

To compare local observation algorithms against the buffered gossip algorithms I described in chapter 4, the local observation algorithms that I study in this dissertation use either maximum frequency selection or proportional selection as their state update protocol. Local observation algorithms that use proportional selection are functionally identical to a voter model [26,27]. Local observation algorithms that use maximum frequency selection are functionally identical to the Label Propagation Algorithm [30, 47].

Mechanics

Local observation algorithms can also be defined by their mechanics. If $u \in V$ is a node that uses a pull-based information propagation algorithm, then node u displays the following behavior: when the clock of node u ticks, node u observes its neighborhood, $N(u)$, and stores the state value of each neighbor in β_u; next, node u updates x_u according to its state update protocol. After x_u has been updated, node u clears its buffer and waits for the next tick of its internal clock. This process of observing neighbors, updating state, and buffer erasing is described by algorithm 3.
If $|W| = 1$ then I call the information propagation algorithm a *simple observation algorithm*. If $1 < |W| \leq |N(u)|$ then I call the information propagation algorithm a *partial observation algorithm*. Finally, if $|W| = |N(u)|$ then I call the information propagation algorithm a *local observation algorithm*. Within the existing literature on pull-based information propagation algorithms for the

Algorithm 3 The Local Observation Algorithm

```
1: procedure ACT(u ∈ V)
2:     for all v ∈ W : W ⊆ N(u) do
3:         β_u ← β_u ∪ (v, x_v)
4:     end for
5:     x_u ← h(f(β_u))
6:     β_u ← ∅
7: end procedure
```

decentralized consensus problem, the majority of work is aimed at the case when $|W| = |N(u)|$. In this dissertation, I continue this trend by focusing on local observation algorithms, and not simple or partial observation algorithms.

Local Observation Algorithms as Solutions to the Decentralized Consensus Problem

Specific types of local observation algorithms, especially the voter model and majority-based algorithms such as the label propagation algorithm, have been long studied in the literature on opinion dynamics and computational social science. It is well known that the the voter model solves the discrete decentralized consensus problem when an asynchronous timing model is used, but not necessarily when the timing model is synchronous [23, 26–28, 52, 53]. The Label Propagation Algorithm is not generally a solution to the discrete decentralized consensus problem [30, 47], but it is possible to sometimes observe consensus formation.

Using the existing knowledge of the voter model and label propagation algorithm, along with the mathematical framework developed for the buffered gossip algorithm, I derive the conditions under which local observation algorithms are capable of solving the decentralized consensus problem in the discrete domain and within the assumptions of this dissertation. Towards this end, I discuss consensus formation in local observation algorithms that use a proportional selection state update protocol, consensus formation in local observation algorithms that use a maximum frequency se-

lection state update protocol, and consensus formation in local observation algorithms in general. I also discuss the impact of noise and error on local observation algorithms.

Consensus under Proportional Selection

A local observation algorithm with a state update protocol that implements proportional selection is a voter model. It is known that consensus will be achieved if a voter model uses an asynchronous timing model [23, 26–28, 52, 53]. Thus, a local observation algorithm that uses an asynchronous timing model with proportional selection will solve the discrete decentralized consensus problem. Alternatively, because proportional selection chooses a single element from a buffer at random, consensus occurs by the same type of arguments as made in lemma 1, lemma 2, and theorem 3. Stability of the consensus is ensured by lemma 4.

Consensus under Maximum Frequency Selection

A local observation protocol with a state update protocol that implements maximum frequency selection is a label propagation algorithm. It is known that the label propagation algorithm is not guaranteed to achieve a consensus when an asynchronous timing model is used. Thus, in general, a local observation algorithm that uses an asynchronous timing model with maximum frequency selection is not guaranteed to solve the discrete decentralized consensus problem.

As an example of this failure to solve the discrete decentralized consensus problem, assume that $G = (V, E)$ is a 4×3 lattice network with $S = \{white, black\}$ and all nodes $u \in V$ are using a local observation algorithm with an asynchronous timing model and a state update protocol that implements maximum frequency selection. Figure 5 displays a configuration under which it is impossible to reach a consensus in G. When the clock of node u_n ticks, u_n will populate β_{u_n} with

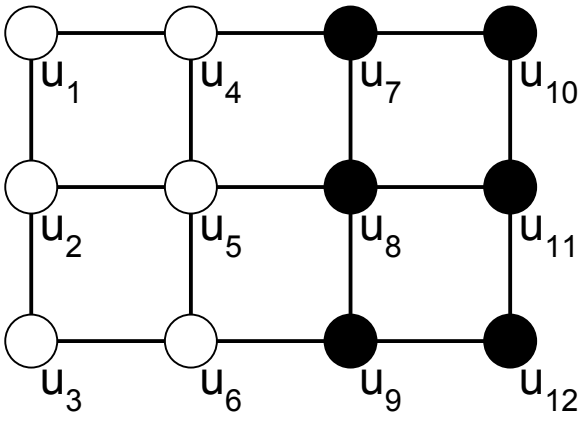

Figure 5.1: A failure scenario for a local observation algorithm using maximum frequency selection.

the state values of its neighbors and then apply maximum frequency selection to pick the new value for x_{u_n}. For nodes u_1, u_2 and u_3 this means that $x_{u_n} = white$ since $u_4 = u_5 = u_6 = white$, while for nodes u_{10}, u_{11} and u_{12} this means that $x_{u_n} = black$ since $u_7 = u_8 = u_9 = black$. Likewise, $x_{u_4} = white$ since $u_1 = u_5 = white$ and $u_7 = black$; $x_{u_5} = white$ since $u_2 = u_4 = u_6 = white$ and $u_8 = black$; and $x_{u_6} = white$ since $u_3 = u_5 = white$ and $u_9 = black$. For similar reasons, $x_{u_n} = black$ for nodes $u_7 = u_8 = u_9$. Thus, it is impossible to achieve a consensus because every node in V will maintain its current state value.

Consensus under Arbitrary Selection-Based State Update Protocols

A local observation algorithm using an arbitrary selection-based state update protocol cannot be guaranteed to solve the discrete decentralized consensus problem because there is a chance that the selection-based state update protocol will exhibit behavior similar to maximum frequency selection. Specifically, because nodes that use local observation algorithms will *always* account for the state value of every neighbor, a directed spanning tree, Ω, in the network G does not neces-

sarily reflect the transmission activity within a system using a local observation algorithm, as it does in the case of a buffered or uniform gossip algorithm. As a consequence of this representation, a consensus sequence may not exist, and so theorem 3 does not always apply. For example, if one assumes that only one node acts per time step, and all nodes have four neighbors, then a node using a local observation will always have four values in its buffer, whereas a node using a buffered gossip algorithm will have at most one value in its buffer. If a buffer has more than one element, then there is no singular "root" node. Therefore, I claim that the viability of a local observation algorithm as a solution to the discrete decentralized consensus problem depends not only the existence of a directed spanning tree, timing model, and state update protocol - but also the neighborhood topology of each node. As an example of this dependency, consider a line graph and a lattice. Nodes using a local observation algorithm on a line graph will reach consensus under maximum frequency selection because the nodes break ties randomly and always have only one or two neighbors, but on a lattice there is a possibility that the fixed state will not represent consensus (see figure 5).

Robustness to Noise and Node Failure

When theorem 3 holds for local observation algorithms, such as in the case of proportional selection, I can conclude that noise will not prevent consensus, but it may interfere with the formation of a consensus sequence and thus reduce the speed at which consensus occurs. The cause for this is the same as with the buffered algorithm. Because the information transmitted between nodes may not be accurate in the presence of noise, partially formed consensus sequences may be broken. However, because noise is random, there is a positive probability that a consensus sequence is able to form without disruption, and so lemma 1 and lemma 2 continue to hold.

I can also conclude that, when theorem 3 holds, node failure will only prevent consensus when two

conditions hold: 1) the node(s) that fail are cut points within every possible directed spanning tree of G; i.e. their removal results in the inability to construct a directed spanning tree in G; and 2) the node(s) that fail never reactivate. These are the same conditions that must occur with a buffered gossip algorithm. If both of these conditions do not hold, then node failure will only delay the formation of a consensus by the same argument given on the impact of noise. Unlike the case of the buffered gossip algorithm, however, systems that use local observation algorithms may be more sensitive to node failure due to their typical buffer size.

An Experimental Comparison between Buffered Gossip and Local Observation Algorithms

Multi-agent simulation can be used to empirically compare the consensus behavior between a set of buffered gossip algorithms, a set of local observation algorithms, and the uniform gossip algorithm. The data from such a comparison can be used to determine whether or buffered gossip and local observation algorithms are similar in their performance on the decentralized consensus problem.

To conduct these comparisons, I use the same simulation software as used in my experiments comparing the buffered gossip algorithm against the uniform gossip algorithm. In the remainder of this section, I describe the experimental design and methodology that structures my experiments. I then discuss my expectations and present the results of my simulations. Finally, I interpret those results as they related to the relative performance of the buffered gossip and local observation algorithms on the consensus problem within a decentralized system of asynchronous agents.

Experimental Design and Methodology

The simulation used to explore the behavior of buffered gossip and local observation algorithms models a multi-agent system consisting of $n = |V|$ asynchronous agents that are connected by a

static communication network. Each node in the network represents an agent and an edge connects two nodes if there is a communication link between the associated agents. The state value of each node represents that node's desired consensus option and is encoded as an integer value. Each node can store up to n transmissions in its buffer, and those transmissions are stored in the order in which they are received. If a node receives multiple transmissions from the same agent before it is able to clear its buffer, only the most recent transmission is retained. To account for the diversity of many real-world networks, the communication network can be structed as either an Erdös-Renyi random network, a Barabasi-Albert scale-free network, a Newman-Watts-Strogatz small world network [50], or a lattice network.

Nodes use an asynchronous timing model, where the expected number of nodes that act in a single time step follows a Poisson distribution with $\lambda = |V|$. Because asynchronous timing models are used, it is possible that some nodes will act multiple times within a single time step. Simulation time is measured in *steps*. One step has passed when all active nodes have updated their state value and spread their information in accordance with their action algorithm. Thus, one step is equivalent to one time step. Those nodes that act within a single step do so in a uniformly random order.

The state update protocol (proportional selection, maximum frequency selection, or tail selection) and the network topology (Erdös-Renyi random, Barabasi-Albert scale-free, Newman-Watts-Strogatz small world, or lattice) are the primary independent variables. For each combination of state update protocol and network topology, I randomly construct 300 networks with the selected topological structure and then conduct 30 independent simulations of rendezvous over each network. These networks are constructed randomly, with $2 \leq |V| \leq 100$ and $1 \leq |S| \leq 5$ being chosen according to a uniform distribution. The decision to vary network and state space size was made to test solution potential over a wide range of possibilities. Additionally, Erdös-Renyi random networks use a random value in the range $[0, 1]$ for their connection probability, and are guaranteed to be connected; Barabase-Albert scale-free networks and Newman-Watts-Strogatz

small world networks are randomly parameterized based on the number of nodes in the network; and lattice networks are guaranteed to be square and do not wrap to form a torus. The parameters associated with each network topology are a requirement of the NetworkX library.

The consensus time (measured in steps) is the dependent variable under study, with the characterization that a value of $100,000$ represents a failure to achieve consensus. Nodes successfully form a consensus if the state of every node is identical within $100,000$ steps. Nodes fail to form a consensus if either periodic behavior is observed or the simulation runs in excess of a maximum time limit ($100,000$ steps). The simulation software is capable of detecting periodic behavior of up to 100 unique states. Behavior is considered to be periodic if a sequence of state distributions repeats continuously for $10,000$ consecutive steps (e.g. a sequence of 10 state distributions repeats $1,000$ times in a row).

Each simulation runs until either consensus is reached, a non-consensus stable state is observed (either fixed or periodic), or a time limit of $110,000$ steps is exceeded. This produces a total of $9,000$ data points per experimental configuration. To remove randomness as a cause for differences between experimental configurations, each configuration is initialized with same sequence of random numbers (i.e. simulation 17 of the configuration $\{$proportional, random$\}$ uses the same random seed as simulation 17 of the configuration $\{$maximum, lattice$\}$).

As with the theoretical proofs of consensus, these experiments focus on information propagation algorithms with proportional selection and maximum frequency selection state update protocols because they are similar to the voter model and the label propagation algorithm; although because they are being used in a new context there is no guarantee that they will display the same behavior. I also make the simplifying assumption that in the event of a node receiving multiple transmissions from the same neighbor prior to a state update, only the most recent transmission is kept in the buffer. Because this dissertation focuses on comparing consensus speed between different

algorithms, and because noise and node failure only prevent consensus formation in very specific scenarios, I assume that information is transmitted without error and nodes do not fail during consensus formation. This assumption simplifies the experiments by holding the noise and node failure probabilities constant at a value of 0.0. Furthermore, I include the uniform gossip algorithm in these experiments as a baseline measure of performance because the uniform gossip algorithm is an established solution for the decentralized consensus problem, and because the uniform gossip algorithm can be implemented as a buffered gossip algorithm. Finally, I rely strictly on graphical analysis and forgo hypothesis testing because there are 5 separate algorithms in these experiments.

Expectations

To compare the relative consensus speed between buffered gossip algorithms, local observations, and the uniform gossip algorithm, I analyze plots and compare basic statistical measures of the data generated by the multi-agent simulation. Based on the known behavior of the voter model and the label propagation algorithm, along with results of my comparison of buffered gossip algorithms against the uniform gossip algorithm, I expect to observe the following behaviors:

- Because a local observation algorithm using maximum frequency selection is functionally equivalent to the label propagation algorithm, and it is known that the label propagation algorithm does not always form a consensus, I expect that the mean consensus time of a buffered gossip algorithm using maximum frequency selection will be less than the mean consensus time of a local observation algorithm using maximum frequency selection.

- Because a local observation algorithm using proportional selection is functionally equivalent to a voter model, and because nodes that use local observation typically have access to more information than nodes that use buffered gossip, I expect that the mean consensus time of a

buffered gossip algorithm using maximum frequency selection will be greater than the mean consensus time of a local observation algorithm using proportional selection.

- Because a local observation algorithm using maximum frequency selection is functionally equivalent to the label propagation algorithm, and it is known that the label propagation algorithm does not always form a consensus, I expect that the mean consensus time of a buffered gossip algorithm using proportional selection will be less than the mean consensus time of a local observation algorithm using maximum frequency selection.

- Because a local observation algorithm using proportional selection is functionally equivalent to a voter model, and because nodes that use local observation typically have access to more information than nodes that use buffered gossip, I expect that the mean consensus time of a buffered gossip algorithm using proportional selection will be greater than the mean consensus time of a local observation algorithm using proportional selection.

- Because a local observation algorithm using maximum frequency selection is functionally equivalent to the label propagation algorithm, and it is known that the label propagation algorithm does not always form a consensus, I expect that the mean consensus time of a uniform gossip algorithm will be less than the mean consensus time of a local observation algorithm using maximum frequency selection.

- Because a local observation algorithm using proportional selection is functionally equivalent to a voter model, and because nodes that use local observation typically have access to more information than nodes that use uniform gossip, I expect that the mean consensus time of a uniform gossip algorithm will be greater than the mean consensus time of a local observation algorithm using proportional selection.

- Because a local observation algorithm using maximum frequency selection is functionally equivalent to the label propagation algorithm, and it is known that the label propagation

algorithm does not always form a consensus, I expect that the mean consensus time of a local observation algorithm using maximum frequency selection will be greater than the mean consensus time of a local observation algorithm using proportional selection.

As with my comparison of buffered gossip algorithms against the uniform gossip algorithm, I also expect to observe differences in consensus speed across network topology.

Empirical Results

This section discusses the results of the experiments on 300 randomly generated Erdös-Renyi random networks, 300 randomly generated Barabasi-Albert scale-free networks, 300 randomly generated Newman-Watts-Strogatz small world networks, and 300 randomly generated lattice networks.

Random Networks

Figure 5.2 visualizes the experimental data from 300 randomly generated Erdös-Renyi random networks using a standard box plot. The upper and lower boundaries of each box correspond to the first and third quartile of the data, with the middle line representing the median value. The upper and lower whiskers extend out to the largest and smallest value within $1.5 \times IRQ$ of the boundary. The individual points represent the outliers of the observed data. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

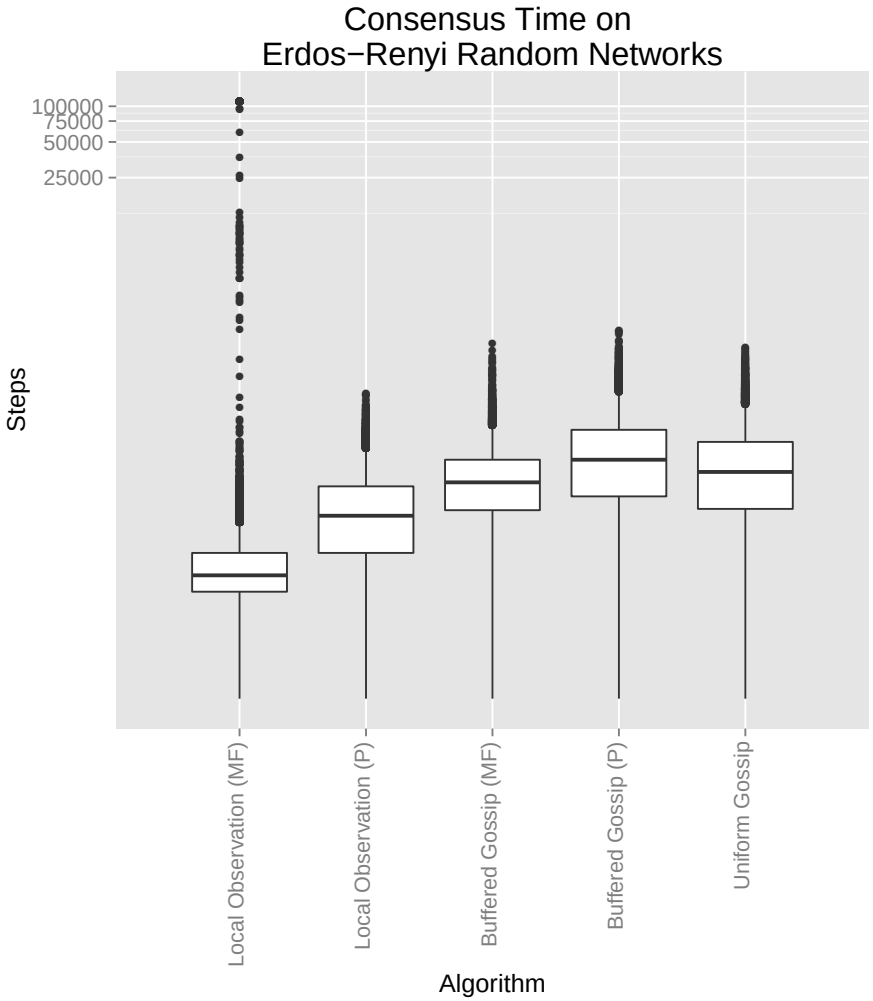

Figure 5.2: Data from the empirical comparison between buffered gossip algorithms and local observation, visualized as box plots that show the interquartile range, median value, and outliers of the consensus time on Erdös-Renyi random networks.

In figure 5.2 one can observe that a local observation algorithm using maximum frequency selection yields the lowest median consensus time and smallest third quartile of the tested algorithms, but there are also many outliers that greatly exceed the worst-case consensus time of the other

information propagation algorithms. These outliers are not unusual given the relationship between a local observation algorithm using maximum frequency selection and the label propagation algorithm. Due to the uncertainty in the performance of a local observation algorithm using maximum frequency selection, a local observation algorithm using proportional selection may yield better results in the long run, even though its typical behavior is worse than a local observation algorithm using maximum frequency selection.

Figure 5.3 visualizes the mean consensus time of the random network data along with the 95% confidence interval of each mean. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

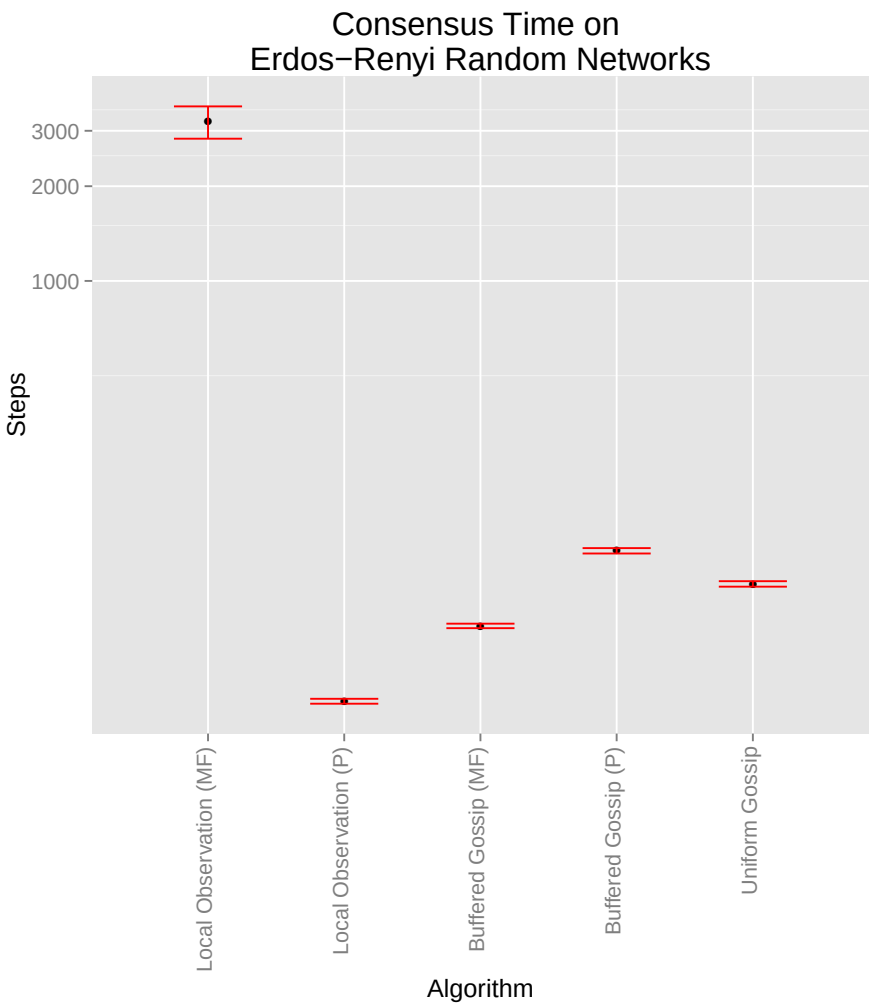

Figure 5.3: Data from the empirical comparison between buffered gossip algorithms and local observation, as errorbar plots of the 95% confidence intervals for the mean consensus time on Erdös-Renyi random networks.

In figure 5.3 one can observe that the mean consensus time of a local observation algorithm using proportional selection ($\mu_{lo.pro} = 45.18, \sigma_{lo.pro} = 41.08$, 95% CI_{μ} $[44.33, 46.03]$) is less than the mean consensus time of a local observation algorithm using maximum frequency se-

lection ($\mu_{lo.max} = 3218.48, \sigma_{lo.max} = 18388.50$, 95% CI_μ $[2838.53, 3598.44]$), a buffered gossip algorithm using either maximum frequency selection ($\mu_{bg.max} = 79.12, \sigma_{bg.max} = 64.94$, 95% CI_μ $[77.77, 80.46]$) or proportional selection ($\mu_{bg.pro} = 137.91, \sigma_{bg.pro} = 128.06$, 95% CI_μ $[135.27, 140.56]$), and the uniform gossip algorithm ($\mu_{uni} = 107.89, \sigma_{uni} = 99.38$, 95% CI_μ $[105.84, 109.95]$).

At the opposite end of the spectrum, a local observation algorithm using maximum frequency selection has a much greater mean consensus time than the compared algorithms. This extreme difference is most likely due to the inclusion of experimental runs in which consensus was not reached, and therefore the consensus time was marked as $110,000$ steps (See figure 5.2). These values were included in the analysis to highlight the impact of failure on the performance of the algorithm. If the simulations had been run for a longer period of time, the associated mean consensus times for failing algorithms would be further biased.

Based on the observations for the experiments on Erdös-Renyi random networks, as represented in figure 5.3 and figure 5.2, I conclude that, with a high probability,

- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is less than the mean consensus time of a local observation algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is greater than the mean consensus time of a local observation algorithm using proportional selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is less than the mean consensus time of a local observation algorithm using maximum frequency selection.

- The mean consensus time of a buffered gossip algorithm using proportional selection is greater than the mean consensus time of a local observation algorithm using proportional selection.
- The mean consensus time of a uniform gossip algorithm is less than the mean consensus time of a local observation algorithm using maximum frequency selection.
- The mean consensus time of a uniform gossip algorithm is greater than the mean consensus time of a local observation algorithm using proportional selection.
- The mean consensus time of a local observation algorithm using maximum frequency selection is greater than the mean consensus time of a local observation algorithm using proportional selection.

In general, the observations represented in figure 5.3 and figure 5.2, suggest that a local observation algorithm using proportional selection is the best choice for fast consensus formation over an arbitrary Erdös-Renyi random network. These observations also suggest that the local observation algorithm using maximum frequency selection is the worst choice for consensus formation due to its lack of stability.

Scale-Free Networks

Figure 5.4 visualizes the experimental data from 300 randomly generated Barabasi-Albert scale-free networks using a standard box plot. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

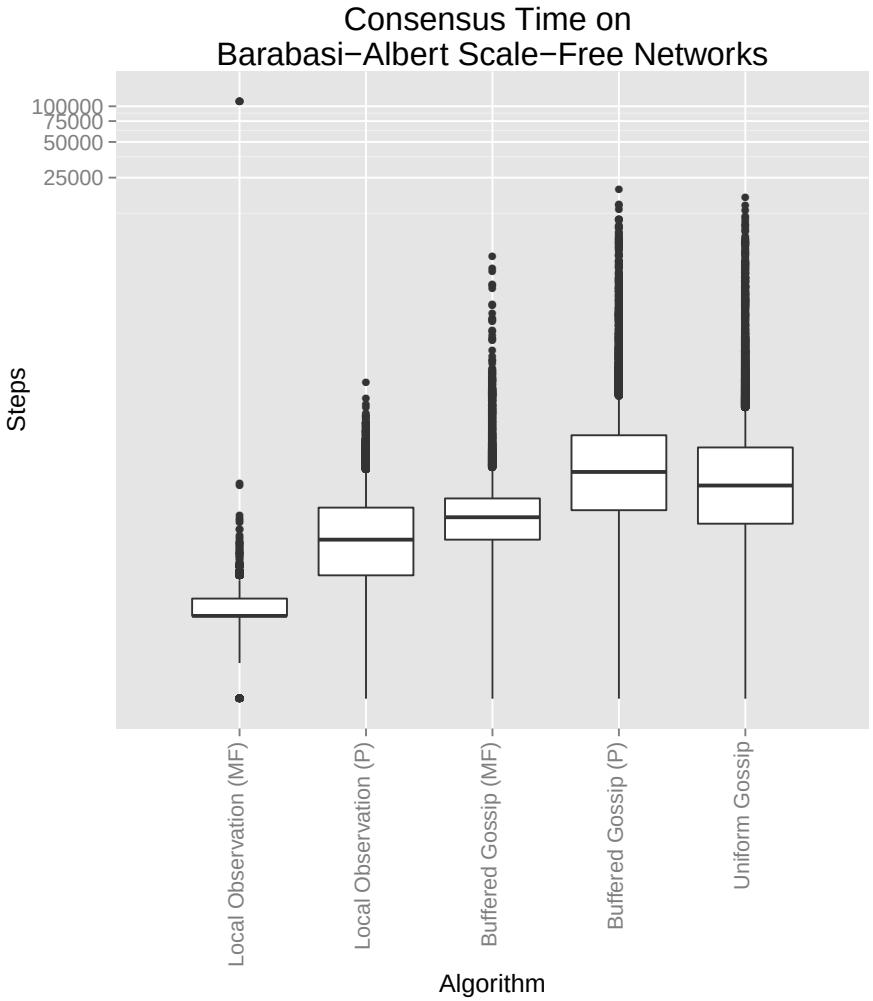

Figure 5.4: Data from the empirical comparison between buffered gossip algorithms and local observation, visualized as box plots that show the interquartile range, median value, and outliers of the consensus time on Barabasi-Albert scale-free networks.

In figure 5.4 one can observe that a local observation algorithm using maximum frequency selection yields the lowest median consensus time and smallest third quartile of the tested algorithms, but there are also outliers that greatly exceed the worst-case consensus time of the other informa-

tion propagation algorithms due to a failure of consensus. If these failed attempts at consensus are accounted for, then a local observation algorithm using proportional selection is expected to have the lowest mean consensus time in the long run.

Figure 5.5 visualizes the mean consensus time of our scale-free network data along with the 95% confidence interval of each mean. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

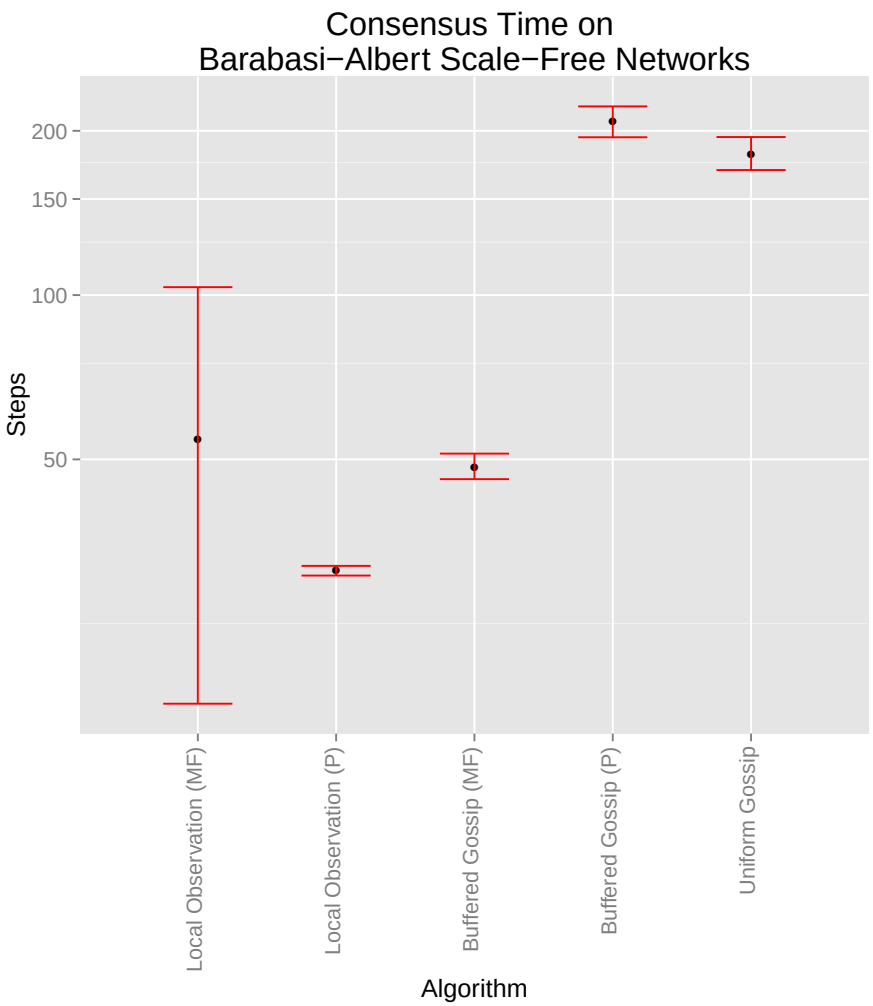

Figure 5.5: Data from the empirical comparison between buffered gossip algorithms and local observation, visualized as errorbar plots of the 95% confidence intervals for the mean consensus time on Barabasi-Albert scale-free networks.

In figure 5.5 one can observe that the mean consensus time of a local observation algorithm using proportional selection ($\mu_{lo.pro} = 30.30, \sigma_{lo.pro} = 30.85$, 95% CI_{μ} $[29.67, 30.94]$) is less than the mean consensus time of a buffered gossip algorithm using either maximum frequency se-

lection ($\mu_{bg.max} = 47.41, \sigma_{bg.max} = 125.68$, 95% CI_μ $[44.81, 50.00]$) or proportional selection ($\mu_{bg.pro} = 206.81, \sigma_{bg.pro} = 692.37$, 95% CI_μ $[192.50, 221.11]$), and the uniform gossip algorithm ($\mu_{uni} = 180.40, \sigma_{uni} = 637.70$, 95% CI_μ $[167.23, 193.58]$). The mean consensus time of a local observation algorithm using maximum frequency selection ($\mu_{lo.max} = 53.51, \sigma_{lo.max} = 2318.54$, 95% CI_μ $[5.61, 101.42]$) is less than a buffered gossip algorithm using proportional selection and less than the uniform gossip algorithm, but not necessarily less than a local observation algorithm using proportional selection or a buffered gossip algorithm using maximum frequency selection.

As with information propagation over a random network, there are instances in which a local observation algorithm using maximum frequency selection fails to achieve consensus. The inclusion of these events biases the mean consensus time and highlights the impact that such failures can have on the performance of the algorithm.

Based on the observations for the experiments on Barabasi-Albert scale-free networks, as represented in figure 5.5 and figure 5.4, I conclude that, with a high probability,

- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is greater than the mean consensus time of a local observation algorithm using proportional selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is not less than the mean consensus time of a local observation algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is greater than the mean consensus time of a local observation algorithm using proportional selection.
- The mean consensus time of a uniform gossip algorithm is not less than the mean consensus

time of a local observation algorithm using maximum frequency selection.

- The mean consensus time of a uniform gossip algorithm is greater than the mean consensus time of a local observation algorithm using proportional selection.

The results of these experiments do not allow me to conclude that "the mean consensus time of a buffered gossip algorithm using maximum frequency selection is less than the mean consensus time of a local observation algorithm using maximum frequency selection." or that "the mean consensus time of a local observation algorithm using maximum frequency selection is greater than the mean consensus time of a local observation algorithm using proportional selection". I conjecture, however, that as the sample size increases, the number of consensus failures under a local observation algorithm with maximum frequency selection will increase the associated mean consensus time until it is much larger than all other algorithms tested in this experiment.

In general, the observations represented in figure 5.5 and figure 5.4, suggest that a local observation algorithm using proportional selection is the best choice for fast consensus formation over an arbitrary Barabasi-Albert scale-free network. These observations also suggest that a buffered gossip algorithm using maximum frequency selection has the highest mean consensus time, however, the local observation algorithm using maximum frequency selection is the worst choice for consensus formation due to its lack of stability.

Small World Networks

Figure 5.6 visualizes the experimental data from 300 randomly generated Newman-Watts-Strogatz small world networks using a standard box plot. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the

data; the data itself has not been transformed.

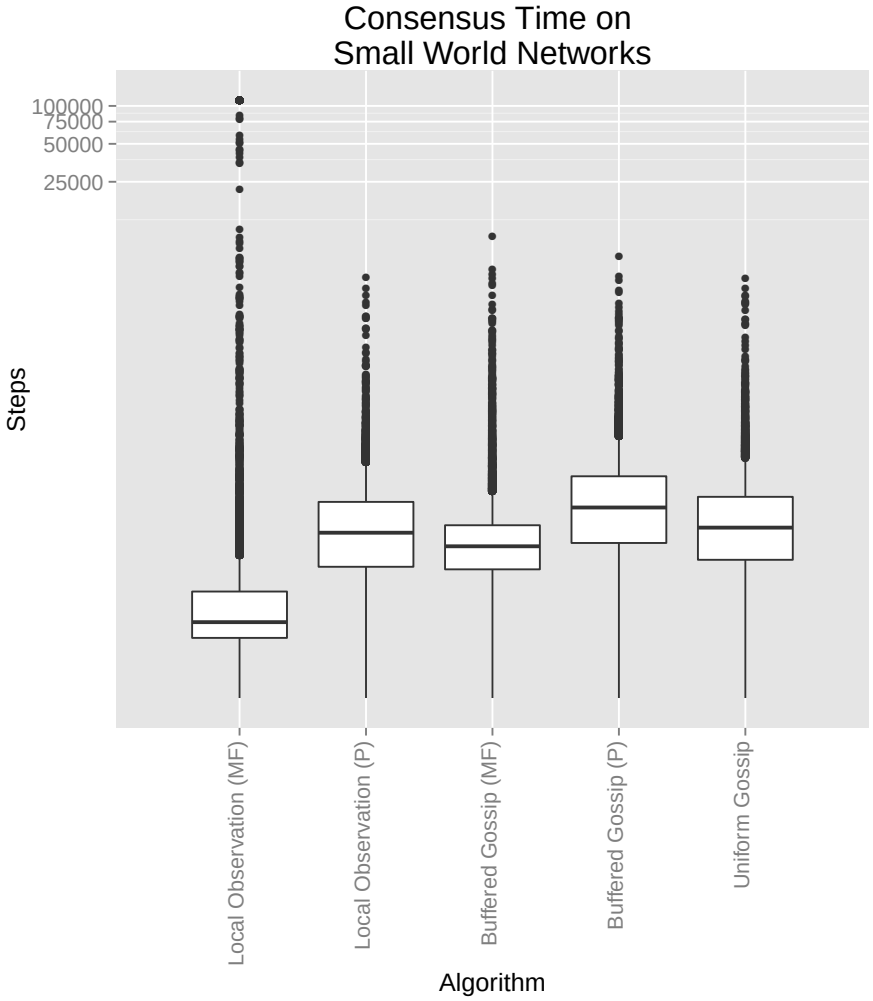

Figure 5.6: Data from the empirical comparison between buffered gossip algorithms and local observation, visualized as box plots that show the interquartile range, median value, and outliers of the consensus time on Newmann-Watts-Strogatz small world networks.

An examine of 5.6 reveals that a local observation algorithm using maximum frequency selection yields the lowest median consensus time and smallest third quartile of the tested algorithms,

but there are also many outliers that greatly exceed the worst-case consensus time of the other information propagation algorithms due to a failure of consensus. If these failed attempts at consensus are accounted for, then either a local observation algorithm using proportional selection or a buffered gossip algorithm using maximum frequency selection is expected to have the lowest mean consensus time in the long run.

Figure 5.7 visualizes the mean consensus time of the small world network data along with the 95% confidence interval of each mean. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

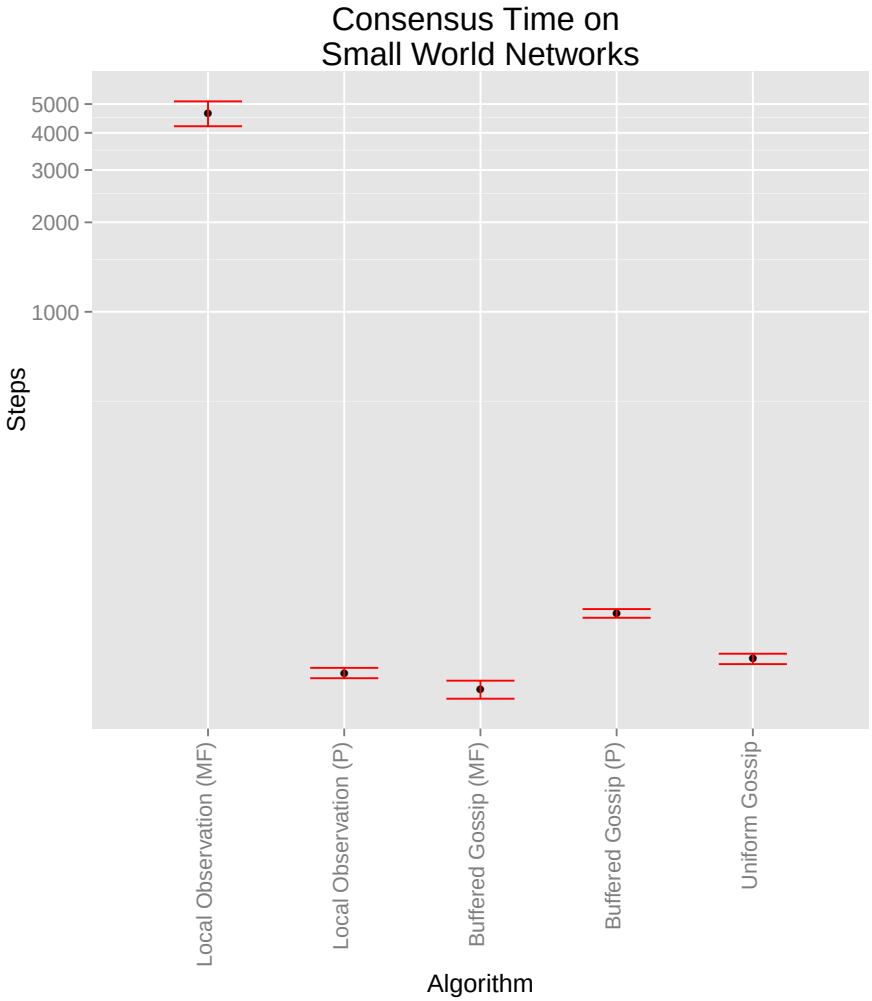

Figure 5.7: Data from the empirical comparison between buffered gossip algorithms and local observation, visualized as errorbar plots of the 95% confidence intervals for the mean consensus time on Newmann-Watts-Strogatz small world networks.

An examination of figure 5.7 reveals that the mean consensus time of a local observation algorithm using proportional selection ($\mu_{lo.pro} = 59.82, \sigma_{lo.pro} = 114.96$, 95% CI_{μ} $[57.45, 62.20]$) is less than the mean consensus time of a buffered gossip algorithm using proportional selec-

tion ($\mu_{bg.pro} = 95.66, \sigma_{bg.pro} = 163.00$, 95% CI_μ $[92.29, 99.02]$) and the uniform gossip algorithm ($\mu_{uni} = 67.07, \sigma_{uni} = 128.73$, 95% CI_μ $[64.41, 69.73]$). It is also much less than the mean consensus time of a local observation algorithm using maximum frequency selection ($\mu_{lo.max} = 4638.21, \sigma_{lo.max} = 21925.25$, 95% CI_μ $[4185.17, 5091.24]$). It is not, however, less than the mean consensus time of a buffered gossip algorithm using maximum frequency selection ($\mu_{bg.max} = 52.49, \sigma_{bg.max} = 180.70$, 95% CI_μ $[48.76, 56.22]$).

As with information propagation over a random network, there are instances in which a local observation algorithm using maximum frequency selection fails to achieve consensus. The inclusion of these events biases the mean consensus time and highlights the impact that such failures can have on the performance of the algorithm. If the simulations had been run for a longer period of time, the associated mean consensus times for failing algorithms would be further biased.

Based on the observations for the experiments on Newman-Watts-Strogatz small world networks, as represented in figure 5.7 and figure 5.6, I conclude that, with a high probability,

- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is less than the mean consensus time of a local observation algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is not greater than the mean consensus time of a local observation algorithm using proportional selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is less than the mean consensus time of a local observation algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is

greater than the mean consensus time of a local observation algorithm using proportional selection.

- The mean consensus time of a uniform gossip algorithm is less than the mean consensus time of a local observation algorithm using maximum frequency selection.
- The mean consensus time of a uniform gossip algorithm is greater than the mean consensus time of a local observation algorithm using proportional selection.
- The mean consensus time of a local observation algorithm using maximum frequency selection is greater than the mean consensus time of a local observation algorithm using proportional selection.

In general, the observations represented in figure 5.7 and figure 5.6, suggest that a buffered gossip algorithm using maximum frequency selection is the best choice for fast consensus formation over an arbitrary Newmann-Watts-Strogatz small world network. These observations also suggest that the local observation algorithm using maximum frequency selection is the worst choice for consensus formation due to its lack of stability.

Lattice Networks

Figure 5.8 visualizes the experimental data from 300 randomly generated lattice networks using a standard box plot. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

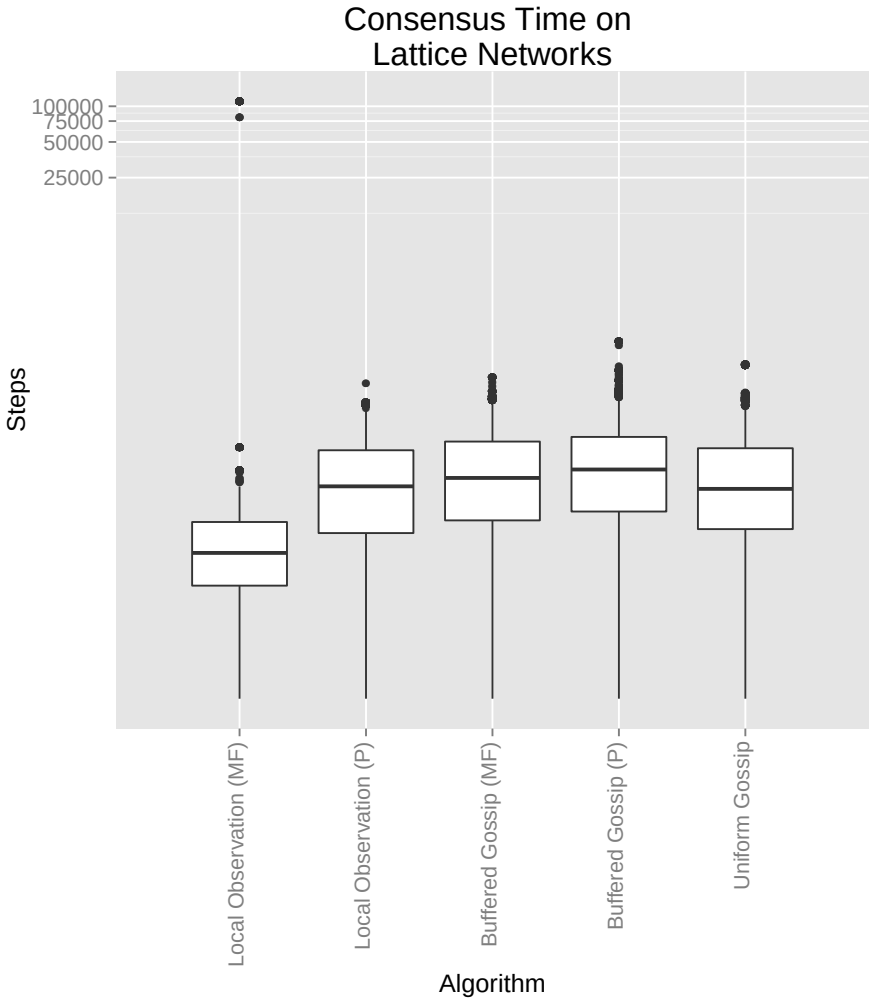

Figure 5.8: Data from the empirical comparison between buffered gossip algorithms and local observation, visualized as box plots that show the interquartile range, median value, and outliers of the consensus time on lattice networks.

An examine of 5.6 reveals that a local observation algorithm using maximum frequency selection yields the lowest median consensus time and smallest third quartile of the tested algorithms, but there are outliers that greatly exceed the worst-case consensus time of the other information propa-

gation algorithms due to a failure of consensus. If these failed attempts at consensus are accounted for, then either a local observation algorithm using proportional selection or the unbuffered gossip algorithm is expected to have the lowest mean consensus time in the long run.

Figure 5.9 visualizes the mean consensus time of the lattice network data along with the 95% confidence interval of each mean. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

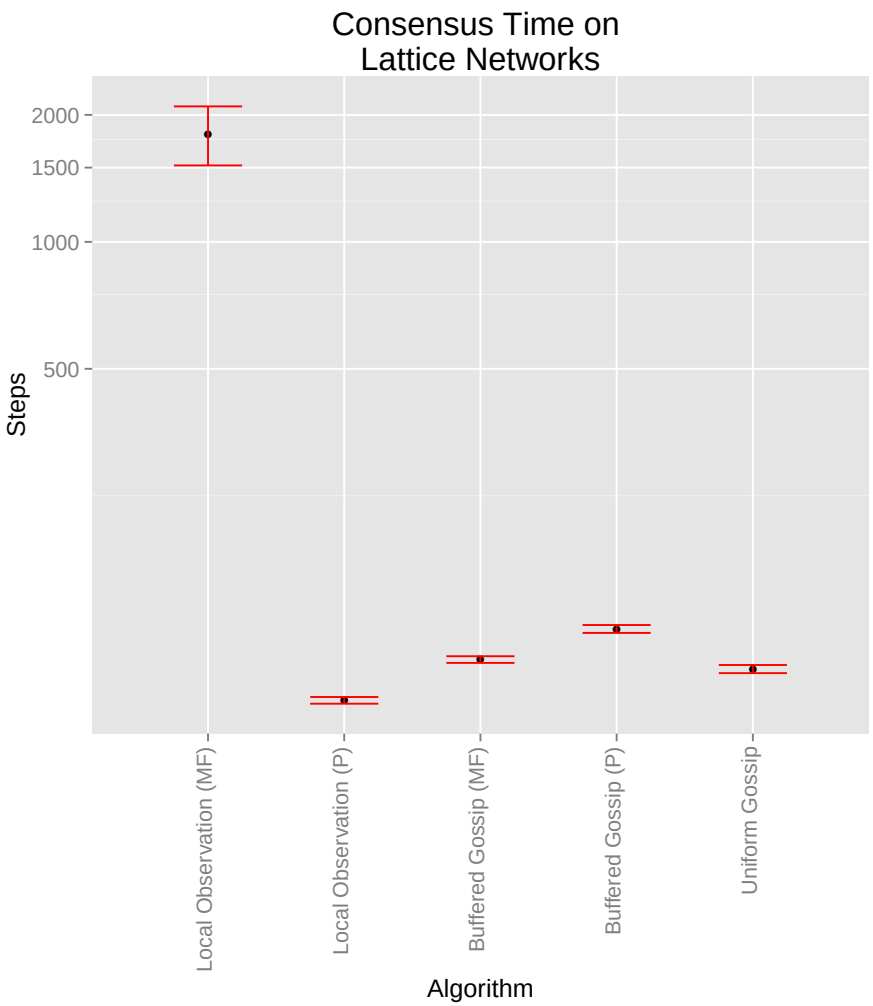

Figure 5.9: Data from the empirical comparison between buffered gossip algorithms and local observation, visualized as errorbar plots of the 95% confidence intervals for the mean consensus time on lattice networks.

In figure 5.9 I observe that the mean consensus time of a local observation algorithm using proportional selection ($\mu_{lo.pro} = 80.78, \sigma_{lo.pro} = 72.10$, 95% CI_{μ} $[79.29, 82.27]$) is less than the mean consensus time of a local observation algorithm using maximum frequency selection ($\mu_{lo.max} =$

$1800.76, \sigma_{lo.max} = 13833.05$, 95% CI_{μ} $[1514.93, 2086.59]$), a buffered gossip algorithm using either maximum frequency selection ($\mu_{bg.max} = 101.26, \sigma_{bg.max} = 92.48$, 95% CI_{μ} $[99.35, 103.17]$) or proportional selection ($\mu_{bg.pro} = 119.69, \sigma_{bg.pro} = 128.80$, 95% CI_{μ} $[117.03, 122.35]$), and the uniform gossip algorithm ($\mu_{uni} = 96.12, \sigma_{uni} = 106.85$, 95% CI_{μ} $[93.91, 98.33]$).

A local observation algorithm using maximum frequency selection has a much greater mean consensus time than the compared algorithms. As with the other experiments related to question Q2, this extreme difference is most likely due to the inclusion of experimental runs in which consensus was not reached (see figure 5.8). If the simulations had been run for a longer period of time, the associated mean consensus times for failing algorithms would be further biased.

Based on the observations for the experiments on lattice networks, as represented in figure 5.9 and figure 5.8, I conclude that, with a high probability,

- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is less than the mean consensus time of a local observation algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is greater than the mean consensus time of a local observation algorithm using proportional selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is less than the mean consensus time of a local observation algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is greater than the mean consensus time of a local observation algorithm using proportional selection.

- The mean consensus time of a uniform gossip algorithm is less than the mean consensus time of a local observation algorithm using maximum frequency selection.
- The mean consensus time of a uniform gossip algorithm is greater than the mean consensus time of a local observation algorithm using proportional selection.
- The mean consensus time of a local observation algorithm using maximum frequency selection is greater than the mean consensus time of a local observation algorithm using proportional selection.

In general, the observations represented in figure 5.9 and figure 5.8, suggest that a local observation algorithm using proportional selection is the best choice for fast consensus formation over an arbitrary lattice network. These observations also suggest that the local observation algorithm using maximum frequency selection is the worst choice for consensus formation due to its lack of stability.

Impact of Network Topology

In regards to comparing the mean consensus time across network topology, the experimental results support the expectation that consensus time is sensitive to the topology of the agent communication network. The communication topology appears to produce a difference in the relative performance of information propagation algorithms using proportional selection or maximum frequency selection, as well as the uniform gossip algorithm.

Summary of Results

Consensus was not observed in all of the experiments comparing buffered gossip algorithms against local observation algorithms. Local observation algorithms using maximum frequency selection failed to arrive at a consensus at least once on every type of network that we tested. All other algorithms, however, did reach consensus.

Unlike with the results of the experiments between buffered gossip algorithms and the uniform gossip algorithm, the topology of the communication network has a lesser impact in the speed of consensus for local observation algorithms than the state update protocol. This is primarily due to the inability of the maximum frequency selection state update protocol to guarantee a consensus. Of particular interest is the observation that local observation algorithms, buffered gossip algorithms, and the uniform gossip algorithm appear to have very similar consensus times on small world and lattice networks. Further research is required to determine if this similarity in performance applies across all regular networks, or just a coincidence.

In regards to the similarity of performance between buffered gossip and local observation algorithms on the decentralized consensus problem, the results suggest that local observation algorithms are typically faster than buffered gossip algorithms - but local observation algorithms using the maximum frequency state update protocol are not guaranteed to produce a consensus.

CHAPTER 6: STOCHASTIC LOCAL OBSERVATION/GOSSIP ALGORITHMS

The local observation algorithms examined in this dissertation tend to yield better performance on the decentralized consensus problem than buffered gossip algorithms, but given identical timing models, transfer protocols, and state update protocols, it cannot be guaranteed that a local observation will always solve the decentralized consensus problem. I use the term *stability* to refer to guarantee that a consensus will be reached, regardless of the length of time required. The buffered gossip algorithms studied in this dissertation are stable. Certain local observation algorithms studied in this dissertation are not stable; e.g. those that use the maximum frequency state update protocol.

I propose Stochastic Local Observation/Gossip (SLOG) algorithms as information propagation algorithm that combines the speed of a local observation algorithm with the stability of a buffered gossip algorithm. This "hybridization" is designed to allow SLOG algorithms to out-perform both buffered gossip algorithms and local observation algorithms on the decentralized consensus problem. To demonstrate this performance improvement, I define SLOG algorithms as information propagation algorithms that use both push-based and pull-based transfer mechanisms. Next, I describe the experimental design and methodology that will be used to empirically compare the consensus behavior between SLOG algorithms, buffered gossip algorithms and local observation algorithms. Finally, I analyze and summarize the results of those experiments.

Stochastic Local Observation/Gossip (SLOG) Algorithms

Stochastic Local Observation/Gossip (SLOG) algorithms combine the speed of a local observation algorithm with the stability of a buffered gossip algorithm. To show how this is possible, I define SLOG algorithms as information propagation algorithms and describe their basic mechanics. Then, I show that SLOG algorithms can successfully solve the decentralized consensus problem when a network contains a directed spanning tree, and the nodes of that network employ both an asynchronous timing model and a selection-based state update protocol.

Definition

As an information propagation algorithm, I define a *Stochastic Local Observation/Gossip* (SLOG) algorithm by fixing the timing model, transfer mechanism, transfer protocol, and state update protocol of node u within a specific range of values.

The timing model of a SLOG algorithm is either asynchronous or synchronous. In this dissertation, I limit my discussion to asynchronous timing models due to the focus on decentralized systems, and because it is often impractical to maintain the synchronization of large decentralized populations.

SLOG algorithms use a stochastic transfer mechanism along with a transfer protocol in which state values are transmitted without modification and stored as part of a tagged pair that identifies both the transmitted state value and identification of the sender. A *stochastic transfer mechanism* combines a uniform gossip transfer mechanism and a local observation transfer mechanism. Nodes that use a stochastic transfer mechanism will act according to a pull-based transfer protocol with probability $0 \leq \alpha \leq 1$ and a push-based transfer mechanism with probability $1 - \alpha$.

As with the other information propagation algorithms that this dissertation studies, nodes that

use a SLOG algorithm use either maximum frequency selection or proportional selection as their state update protocol. SLOG algorithms that use proportional selection can be thought of as voter models on dynamic networks - they randomly consider every neighbor with probability α and a subset of their entire neighborhood with probability $1 - \alpha$. Similarly, SLOG algorithms that use maximum frequency selection can be thought of as Label Propagation Algorithms on dynamic networks.

Mechanics

SLOG algorithms can also be defined by their mechanics. A *stochastic push-pull information propagation algorithm* is an information propagation algorithm in which nodes use a stochastic transfer mechanism. If $u \in V$ is a node that uses a stochastic push-pull information propagation algorithm, then node u displays the following behavior: when the internal clock of node u ticks, the first thing that node u does is to determine whether or not, with uniform probability α, it will clear its buffer and replace the contents with the observed values of one or more neighbors; next, node u updates its state value according to its state update protocol; following this update, the new state value of node u is transmitted to and stored in the buffer of one or more randomly chosen neighbors; after transmission has occurred, node u clears its buffer and waits for the next tick of its internal clock. This process of stochastic observation, updating state, transmitting from node u to neighbors $W \subseteq N(u)$, and buffer erasing is described by algorithm 4. The size of W and Y determine the specific stochastic push-pull information propagation algorithm that is implemented. As with push-based and pull-based information propagation algorithms, $|W| = 1$, $1 < |W| \leq |N(u)|$, or $|W| = |N(u)|$ and likewise for Y. In order to remain consistent with the buffered gossip and local observation algorithms invested in this current body of research, I study the case when $|W| = |N(u)|$ and $|Y| = 1$.

Algorithm 4 The Stochastic Local Observation/Gossip Algorithm

```
1: procedure Act(u ∈ V)
2:     X ← UNIFORM(0, 1)
3:     if X < α then
4:         β_u = ∅
5:         for all v ∈ W : W ⊆ N(u) do
6:             β_u ← β_u ∪ (v, x_v)
7:         end for
8:     end if
9:     x_u ← h(f(β_u))
10:    for all v ∈ Y : Y ⊆ N(u) do
11:        β_v ← β_v ∪ (u, x_u)
12:    end for
13:    β_u ← ∅
14: end procedure
```

SLOG Algorithms as Solutions to the Consensus Problem

The first step in showing that the SLOG algorithm is a solution to the consensus problem is to identify the values of $alpha$ for which consensus formation is guaranteed. The second step is to identify the conditions under which the SLOG algorithm is robust to noise and error.

The conditions under which a SLOG algorithm solves the decentralized consensus problem are trivial when $\alpha = 0$ or $\alpha = 1$.

Lemma 5 *If a finite network, G, contains a directed spanning tree and if the nodes in G use a SLOG algorithm with $\alpha = 0$, a selection-based state update protocol and an asynchronous timing model, then a stable consensus will be obtained in asymptotic time.*

Proof 5 *If $\alpha = 0$ then a SLOG algorithm is a buffered gossip algorithm, and hence by direct application of theorem 3 and 4 it is a solution to the consensus problem.* ■

Lemma 6 *If a finite network, G, contains a directed spanning tree and if the nodes in G use a*

SLOG algorithm with $\alpha = 1$, a selection-based state update protocol and an asynchronous timing model, then a stable consensus is not guaranteed to be obtained in asymptotic time.

Proof 6 *If $\alpha = 1$ then a SLOG algorithm is an observation algorithm. A counterexample to successful consensus formation when maximum frequency selection is used as the state update protocol is given in figure 5, and so it cannot be guaranteed that a SLOG algorithm will solve the consensus problem for an arbitrary state update protocol.* ■

When $0 < \alpha < 1$ we can show when the SLOG algorithm solves the discrete decentralized consensus problem using a technique similar to the one we used to show that the buffered gossip algorithm solves the discrete decentralized consensus problem.

Theorem 7 *If a finite network, G, contains a directed spanning tree and if the nodes in G use a SLOG algorithm with $0 < \alpha < 1$, a selection-based state update protocol and an asynchronous timing model, then a stable consensus will be obtained in asymptotic time.*

Proof 7 *If $0 < \alpha < 1$ then with probability $1 - \alpha > 0$ a SLOG algorithm acts as a buffered gossip algorithm at tick t. Thus, there is a positive probability that a SLOG algorithm acts continuously as a buffered gossip algorithm long enough to construct a consensus sequence as per lemma 1. So, by the second Borel-Cantelli Lemma, a consensus sequence is observed with probability 1 as $t \to \infty$. Hence, in a network where all nodes use a SLOG algorithm, a stable consensus will be achieved in asymptotic time as per theorem 3 and lemma 4.* ■

Thus, a SLOG algorithm is guaranteed to produce a consensus when $0 \leq \alpha < 1$. A consensus is possible when $\alpha = 1$, but the specific state update protocol will determine if it is guaranteed.

With regards to robustness to noise and error, SLOG algorithms are also subject to the same behaviors as buffered gossip algorithms and local observation algorithms. When $\alpha = 1$, a SLOG algorithm is functionally identical to a local observation algorithm. When $\alpha = 1$, a SLOG algorithm is functionally identical to a buffered gossip algorithm. In the case where $1 < \alpha < 0$, the same arguments about robustness can made as for consensus formation – SLOG algorithms behave like buffered gossip algorithms.

An Experimental Investigation of the SLOG Algorithm

Multi-agent simulation can be used to empirically explore a SLOG algorithm and compare it's consensus behavior against buffered gossip algorithms, local observation algorithms, and the uniform gossip algorithm. The data from these comparisons can be used to determine the relative performance of each algorithm on the decentralized consensus problem and test whether or not SLOG algorithms really do out-perform other information propagation algorithms with the same timing model, transfer protocol, and state update protocol.

To conduct these comparisons, I use the same simulation software as used in my experiments comparing the buffered gossip algorithm against the uniform gossip algorithm. In the remainder of this section, I describe the experimental design and methodology that structures my experiments. I then discuss my expectations and present the results of my simulations. Finally, I interpret those results as they related to the relative performance of the buffered gossip and local observation algorithms on the consensus problem within a decentralized system of asynchronous agents.

Experimental Design and Methodology

The simulation used to explore the behavior of SLOG algorithms and compare them against buffered gossip algorithms, local observation algorithms, and the uniform gossip algorithm models a multi-agent system of $n = |V|$ asynchronous agents that are connected by a static communication network. Each node in the network represents an agent and an edge connects two nodes if there is a communication link between the associated agents. The state value of each node represents that node's desired consensus option and is encoded as an integer value. Each node can store up to n transmissions in its buffer, and those transmissions are stored in the order in which they are received. If a node receives multiple transmissions from the same agent before it is able to clear its buffer, only the most recent transmission is retained. To account for the diversity of many real-world networks, the communication network can be structed as either an Erdös-Renyi random network, a Barabasi-Albert scale-free network, a Newman-Watts-Strogatz small world network [50], or a lattice network.

Nodes use an asynchronous timing model, where the expected number of nodes that act in a single time step follows a Poisson distribution with $\lambda = |V|$. Because asynchronous timing models are used, it is possible that some nodes will act multiple times within a single time step. Simulation time is measured in *steps*. One step has passed when all active nodes have updated their state value and spread their information in accordance with their action algorithm. Thus, one step is equivalent to one time step. Those nodes that act within a single step do so in a uniformly random order.

The state update protocol (proportional selection, maximum frequency selection, or tail selection) and the network topology (Erdös-Renyi random, Barabasi-Albert scale-free, Newman-Watts-Strogatz small world, or lattice) are the primary independent variables. For each combination of state update protocol and network topology, I randomly construct 300 networks with the selected topological structure and then conduct 30 independent simulations of rendezvous over each

network. These networks are constructed randomly, with $2 \leq |V| \leq 100$ and $1 \leq |S| \leq 5$ being chosen according to a uniform distribution. The decision to vary network and state space size was made to test solution potential over a wide range of possibilities. Additionally, Erdös-Renyi random networks use a random value in the range $[0, 1]$ for their connection probability, and are guaranteed to be connected; Barabase-Albert scale-free networks and Newman-Watts-Strogatz small world networks are randomly parameterized based on the number of nodes in the network; and lattice networks are guaranteed to be square and do not wrap to form a torus. The parameters associated with each network topology are a requirement of the NetworkX library.

The consensus time (measured in steps) is the dependent variable under study, with the characterization that a value of $100,000$ represents a failure to achieve consensus. Nodes successfully form a consensus if the state of every node is identical within $100,000$ steps. Nodes fail to form a consensus if either periodic behavior is observed or the simulation runs in excess of a maximum time limit ($100,000$ steps). The simulation software is capable of detecting periodic behavior of up to 100 unique states. Behavior is considered to be periodic if a sequence of state distributions repeats continuously for $10,000$ consecutive steps (e.g. a sequence of 10 state distributions repeats $1,000$ times in a row).

Each simulation runs until either consensus is reached, a non-consensus stable state is observed (either fixed or periodic), or a time limit of $110,000$ steps is exceeded. This produces a total of $9,000$ data points per experimental configuration. To remove randomness as a cause for differences between experimental configurations, each configuration is initialized with same sequence of random numbers (i.e. simulation 17 of the configuration $\{$proportional, random$\}$ uses the same random seed as simulation 17 of the configuration $\{$maximum, lattice$\}$).

As with the theoretical proofs of consensus, these experiments focus on information propagation algorithms with proportional selection and maximum frequency selection state update protocols

because they are similar to the voter model and the label propagation algorithm; although because they are being used in a new context there is no guarantee that they will display the same behavior. I also make the simplifying assumption that in the event of a node receiving multiple transmissions from the same neighbor prior to a state update, only the most recent transmission is kept in the buffer. Because this dissertation focuses on comparing consensus speed between different algorithms, and because noise and node failure only prevent consensus formation in very specific scenarios, I assume that information is transmitted without error and nodes do not fail during consensus formation. This assumption simplifies the experiments by holding the noise and node failure probabilities constant at a value of 0.0. Finally, I rely strictly on graphical analysis and forgo hypothesis testing because there are 7 separate algorithms in these experiments.

Expectations

To compare the relative consensus speed between information propagation algorithms, I analyze plots and compare basic statistical measures of the data generated by the multi-agent simulation. Because SLOG algorithms combine local observation with buffered gossip, it is expected that their performance will lie between local observation algorithms and buffered gossip algorithms; however, because local observation algorithms using maximum frequency selection do not always solve the consensus problem, I expect to observe the following results:

- The mean consensus time of a buffered gossip algorithm using maximum frequency selection will be greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using maximum frequency selection will be greater than the mean consensus time of a SLOG algorithm using proportional selection.

- The mean consensus time of a buffered gossip algorithm using proportional selection will be greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection will be greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a uniform gossip algorithm will be greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a uniform gossip algorithm will be greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a local observation algorithm using maximum frequency selection will be greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a local observation algorithm using maximum frequency selection will be greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a local observation algorithm using proportional selection will be greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a local observation algorithm using proportional selection will be less than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a SLOG algorithm using maximum frequency selection will be less than the mean consensus time of a SLOG algorithm using proportional selection.

As with my comparison of other information propagation algorithms, I also expect to observe differences in consensus speed across network topology.

Empirical Results

This section discusses the results of the experiments on 300 randomly generated Erdös-Renyi random networks, 300 randomly generated Barabasi-Albert scale-free networks, 300 randomly generated Newman-Watts-Strogatz small world networks, and 300 randomly generated lattice networks are discussed below.

Random Networks

Figure 6.1 visualizes the experimental data from 300 randomly generated Erdös-Renyi random networks using a standard box plot. The upper and lower boundaries of each box correspond to the first and third quartile of the data, with the middle line representing the median value. The upper and lower whiskers extend out to the largest and smallest value within $1.5 \times IRQ$ of the boundary. The individual points represent the outliers of the observed data. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

In figure 6.1, I observe that a local observation algorithm using maximum frequency selection and a SLOG algorithm using maximum frequency selection have a similar median consensus time and a similar interquartile range for the consensus time. Furthermore, these values appear to be less than the corresponding values of the other five algorithms considered in this experiment. The outlying data points associated with the SLOG algorithm, however, are much less than those as-

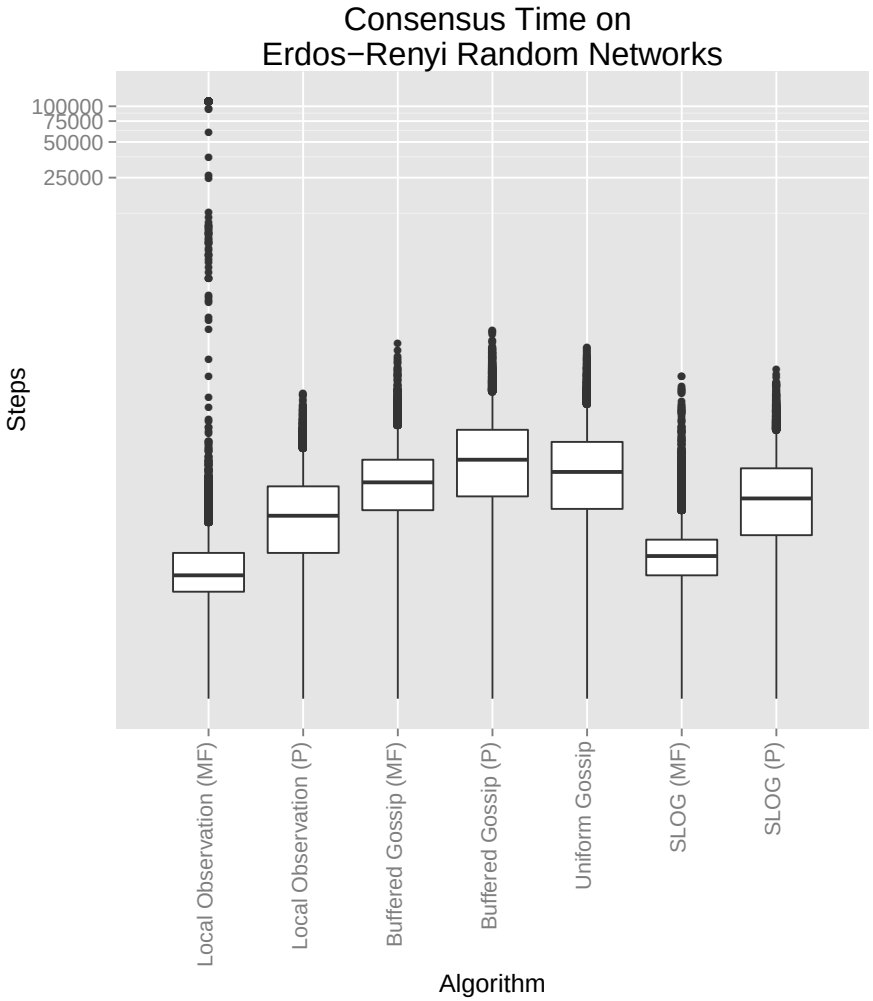

Figure 6.1: Data from the empirical comparison between information propagation algorithms, visualized as box plots that show the interquartile range, median value, and outliers of the consensus time on Erdös-Renyi random networks.

sociated with the local observation algorithm. This observation suggests that of the seven total algorithms compared, a Stochastic Local Observation/Gossip algorithm should tend to offer the fastest consensus speed.

Figure 6.2 visualizes the mean consensus time of the random network data along with the 95%

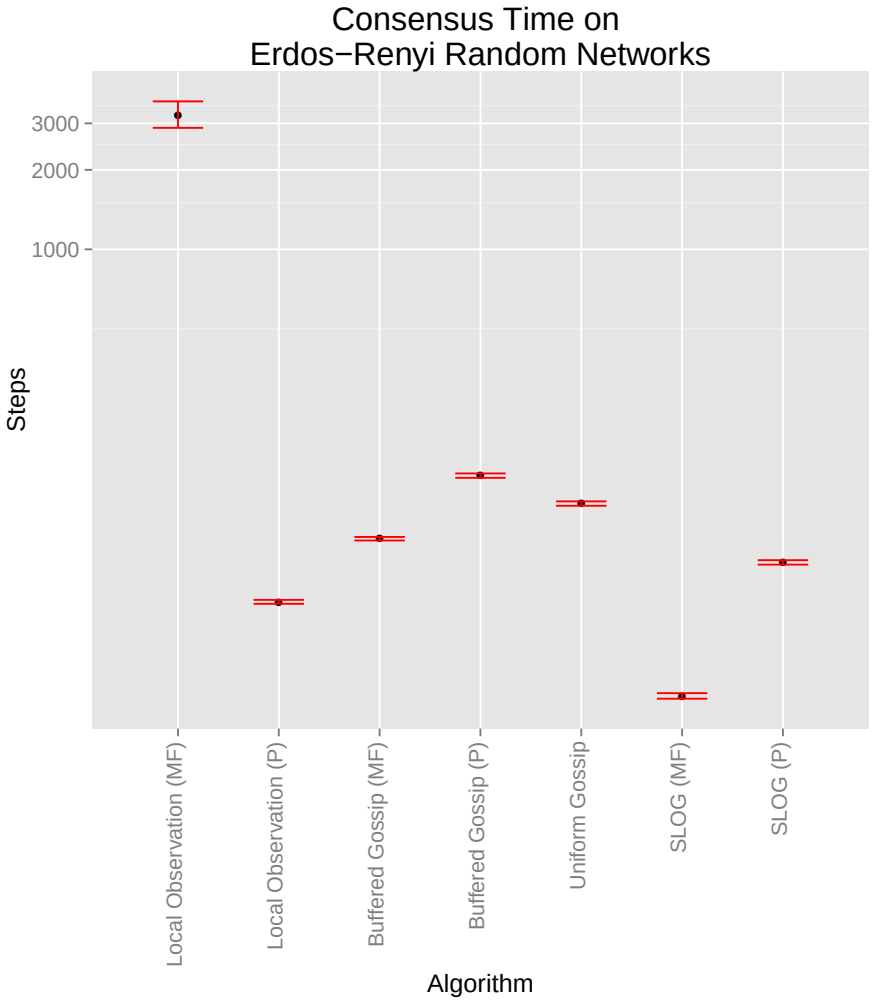

Figure 6.2: Data from the empirical comparison between information propagation algorithms,visualized as errorbar plots of the 95% confidence intervals for the mean consensus time on Erdös-Renyi random networks.

confidence interval of each mean. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

In figure 6.2, one can observe that the mean consensus time of a SLOG algorithm with maximum frequency selection ($\mu_{slog.max} = 19.32, \sigma_{slog.max} = 23.98$, 95% CI_μ $[18.82, 19.82]$) is less than all other algorithms studied in this experiment. The mean consensus time of a local observation algorithm using maximum frequency selection is the highest ($\mu_{lo.max} = 3218.48, \sigma_{lo.max} = 18388.50$, 95% CI_μ $[2838.53, 3598.44]$). The mean consensus time of a SLOG algorithm using proportional selection ($\mu_{slog.pro} = 64.18, \sigma_{slog.pro} = 59.20$, 95% CI_μ $[62.96, 65.40]$) is less than the mean consensus time of buffered gossip algorithms using either maximum frequency selection ($\mu_{bg.max} = 79.12, \sigma_{bg.max} = 64.94$, 95% CI_μ $[77.77, 80.46]$) or proportional selection ($\mu_{bg.pro} = 137.91, \sigma_{bg.pro} = 128.06$, 95% CI_μ $[135.27, 140.56]$), and is also less than the mean consensus time of the uniform gossip algorithm ($\mu_{uni} = 107.89, \sigma_{uni} = 99.38$, 95% CI_μ $[105.84, 109.95]$). The mean consensus time of a local observation algorithm using proportional selection ($\mu_{lo.pro} = 45.18, \sigma_{lo.pro} = 41.08$, 95% CI_μ $[44.33, 46.03]$), however, is less than a SLOG algorithm with proportional selection.

Based on the observations for the experiments on Erdös-Renyi random networks, as represented in figure 6.2 and figure 6.1, I conclude that, with a high probability,

- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.

- The mean consensus time of a buffered gossip algorithm using proportional selection is greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a uniform gossip algorithm is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a uniform gossip algorithm is greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a local observation algorithm using maximum frequency selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a local observation algorithm using maximum frequency selection is greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a local observation algorithm using proportional selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a local observation algorithm using proportional selection is less than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a SLOG algorithm using maximum frequency selection is less than the mean consensus time of a SLOG algorithm using proportional selection.

In general, the observations represented in figure 6.2 and figure 6.1, suggest that a SLOG algorithm using maximum frequency selection is the best choice for fast consensus formation over an arbitrary Erdös-Renyi random network. These observations also suggest that the local observation

algorithm using maximum frequency selection is the worst choice for consensus formation due to its lack of stability.

Scale-Free Networks

Figure 6.3 visualizes the experimental data from 300 randomly generated Barabasi-Albert scale-free networks using a standard box plot. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

In figure 6.3, one can observe that a local observation algorithm using maximum frequency selection has a slightly smaller median consensus time than a SLOG algorithm using maximum frequency selection, but a similar interquartile range for the consensus time. Furthermore, these values appear to be less than the corresponding values of the other five algorithms considered in this experiment. The outlying data points associated with the SLOG algorithm, however, are much less than those associated with the local observation algorithm (which has runs in which consensus is not achieved, as denoted by the outliers with a value of $110,000$). This observation suggests that of the seven total algorithms compared, a Stochastic Local Observation/Gossip algorithm should tend to offer the fastest consensus speed.

Figure 6.4 visualizes the mean consensus time of our scale-free network data along with the 95% confidence interval of each mean. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

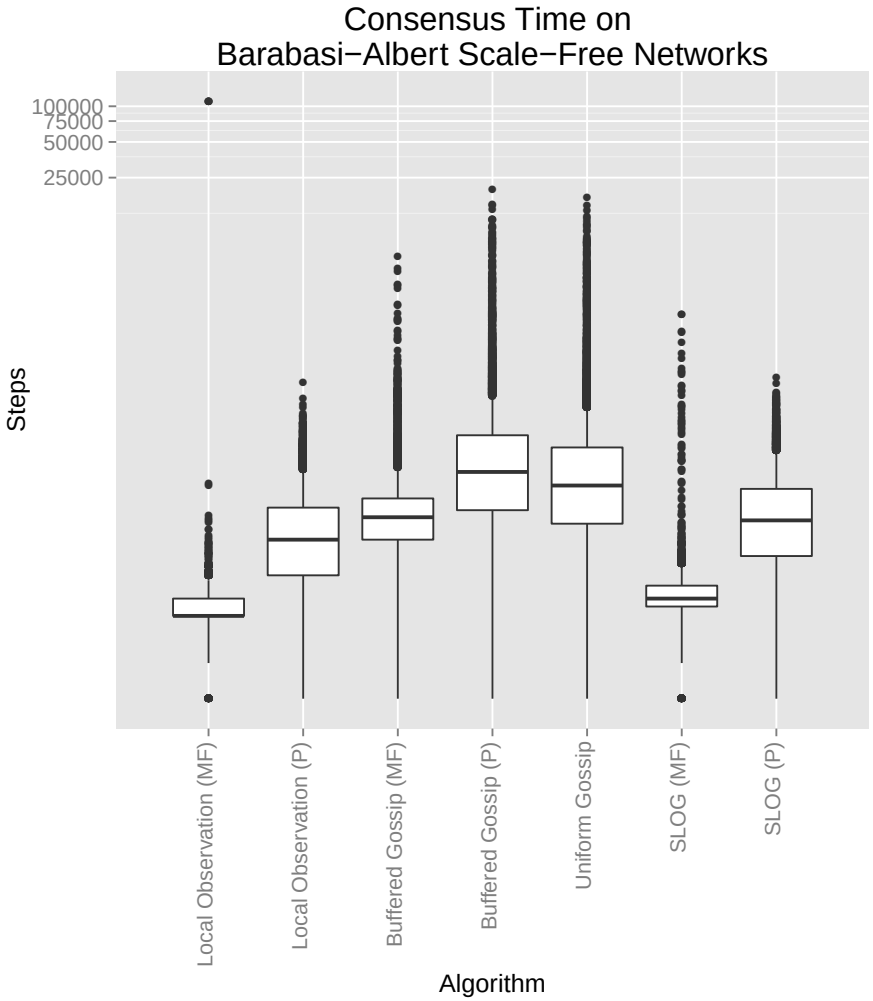

Figure 6.3: Data from the empirical comparison between information propagation algorithms, visualized as box plots that show the interquartile range, median value, and outliers of the consensus time on Barabasi-Albert scale-free networks.

In figure 6.4, one can observe that the mean consensus time of a SLOG algorithm with maximum frequency selection ($\mu_{slog.max} = 8.53, \sigma_{slog.max} = 40.06$, 95% CI_{μ} $[7.71, 9.36]$) is less than all other algorithms studied in this experiment. The mean consensus time of a buffered gossip algorithm algorithm using proportional selection is the highest ($\mu_{bg.pro} = 206.81, \sigma_{bg.pro} = 692.37$, 95% CI_{μ} $[192.50, 221.11]$). The mean consensus time of a SLOG algorithm using proportional

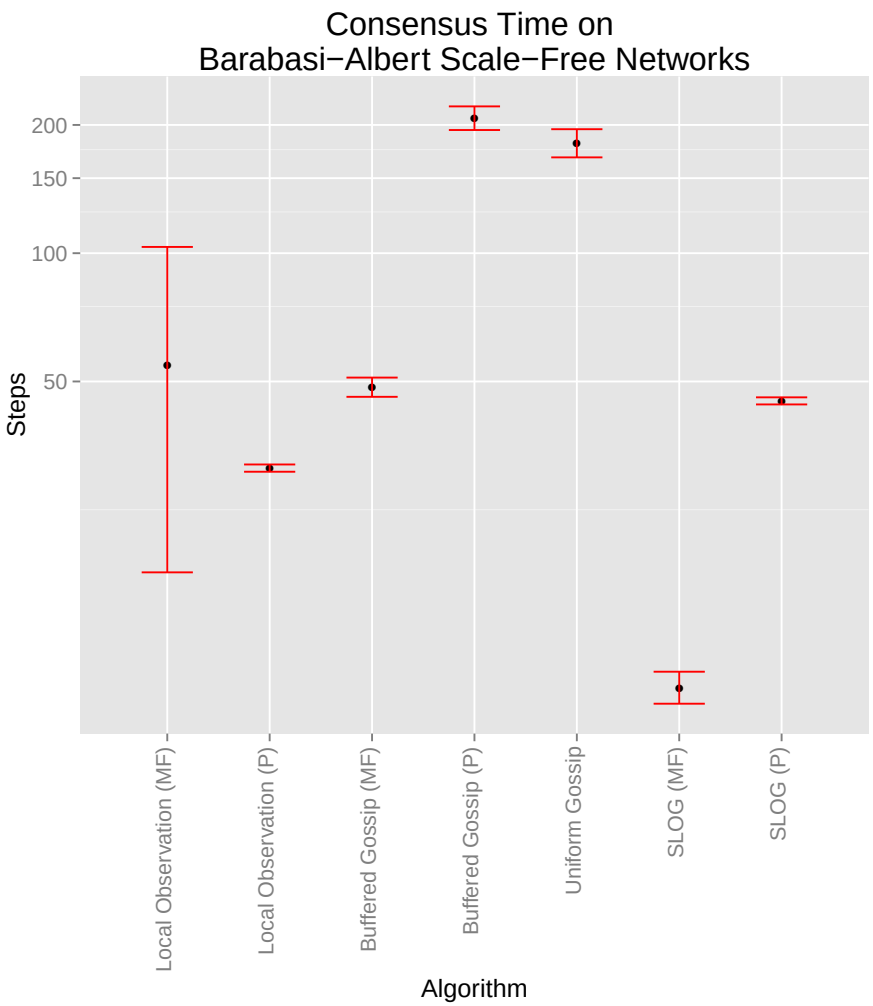

Figure 6.4: Data from the empirical comparison between information propagation algorithms, visualized as errorbar plots of the 95% confidence intervals for the mean consensus time on Barabasi-Albert scale-free networks.

selection ($\mu_{slog.pro} = 44.00, \sigma_{slog.pro} = 43.59$ 95%, CI_μ $[43.10, 44.90]$) is less than the mean consensus time of buffered gossip algorithms using either maximum frequency selection ($\mu_{bg.max} = 47.41, \sigma_{bg.max} = 125.68$, 95% CI_μ $[44.81, 50.00]$) or proportional selection, and is also less than the mean consensus time of the uniform gossip algorithm ($\mu_{uni} = 180.40, \sigma_{uni} = 637.70$, 95% CI_μ $[167.23, 193.58]$). The mean consensus time of a local observation algorithm using pro-

portional selection ($\mu_{slog.pro} = 44.00, \sigma_{slog.pro} = 43.59$, 95% CI_μ $[43.10, 44.90]$), however, is less than a SLOG algorithm with proportional selection. The relative comparison of the mean consensus time for a local observation algorithm using maximum frequency selection is inconclusive due to the scenarios in which the algorithm fails to produce a consensus on a scale-free network. In 6.4 this is manifest by the large confidence interval for the associated algorithm ($\mu_{lo.max} = 53.51, \sigma_{lo.max} = 2318.54$, 95% CI_μ $[5.61, 101.42]$), but in reality, this interval will continue to widen as the sample size increases.

Based on the observations for the experiments on Barabasi-Albert scale-free networks, as represented in figure 6.4 and figure 6.3, I conclude that, with a high probability,

- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a uniform gossip algorithm is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a uniform gossip algorithm is greater than the mean consensus time of a SLOG algorithm using proportional selection.

- The mean consensus time of a local observation algorithm using maximum frequency selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a local observation algorithm using proportional selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a local observation algorithm using proportional selection is less than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a SLOG algorithm using maximum frequency selection is less than the mean consensus time of a SLOG algorithm using proportional selection.

These results do not allow me to conclude that "the mean consensus time of a local observation algorithm using maximum frequency selection will be greater than the mean consensus time of a SLOG algorithm using proportional selection".

In general, the observations represented in figure 6.4 and figure 6.3, suggest that a SLOG algorithm using maximum frequency selection is the best choice for fast consensus formation over an arbitrary Barabasi-Albert scale-free network. These observations also suggest that a buffered gossip algorithm using maximum frequency selection has the highest mean consensus time, however, the local observation algorithm using maximum frequency selection is the worst choice for consensus formation due to its lack of stability.

Small World Networks

Figure 6.5 visualizes the experimental data from 300 randomly generated Newman-Watts-Strogatz small world networks using a standard box plot. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

In figure 6.5, one can observe that a SLOG algorithm using maximum frequency selection and a local observation algorithm using maximum frequency selection have similar smaller median consensus times along with a similar interquartile ranges for the consensus time. Furthermore, these values appear to be less than the corresponding values of the other five algorithms considered in this experiment. The outlying data points associated with the SLOG algorithm, however, are much less than those associated with the local observation algorithm (which has runs in which consensus is not achieved, as denoted by the outliers with a value of $110,000$). This observation suggests that of the seven total algorithms compared, a Stochastic Local Observation/Gossip algorithm should tend to offer the fastest consensus speed.

Figure 6.6 visualizes the mean consensus time of the small world network data along with the 95% confidence interval of each mean. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

In figure 6.6, one can observe that the mean consensus time of a SLOG algorithm with maximum frequency selection ($\mu_{slog.max} = 28.36, \sigma_{slog.max} = 125.35$, 95% CI_{μ} $[25.77, 30.95]$) is less than all other algorithms studied in this experiment. The mean consensus time of a local observation algo-

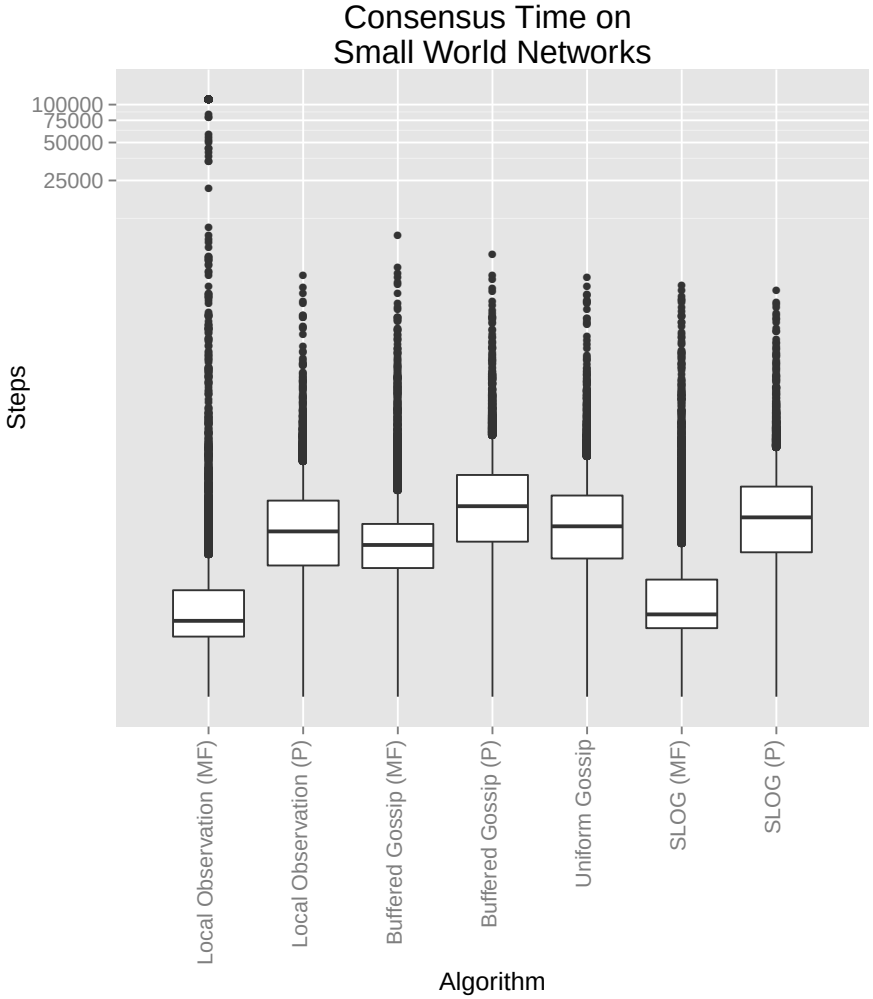

Figure 6.5: Data from the empirical comparison between information propagation algorithms, visualized as box plots that show the interquartile range, median value, and outliers of the consensus time on Newmann-Watts-Strogatz small world networks.

rithm using maximum frequency selection is the highest ($\mu_{lo.max} = 4638.21, \sigma_{lo.max} = 21925.25$, 95% CI_μ $[4185.17, 5091.24]$). The mean consensus time of a SLOG algorithm using proportional selection ($\mu_{slog.pro} = 75.51, \sigma_{slog.pro} = 112.61$, 95% CI_μ $[73.18, 77.83]$) is less than the mean consensus time of buffered gossip algorithms using proportional selection ($\mu_{bg.pro} = 95.66, \sigma_{bg.pro} = 163.00$, 95% CI_μ $[92.29, 99.02]$), but greater than the mean consensus time of a buffered gos-

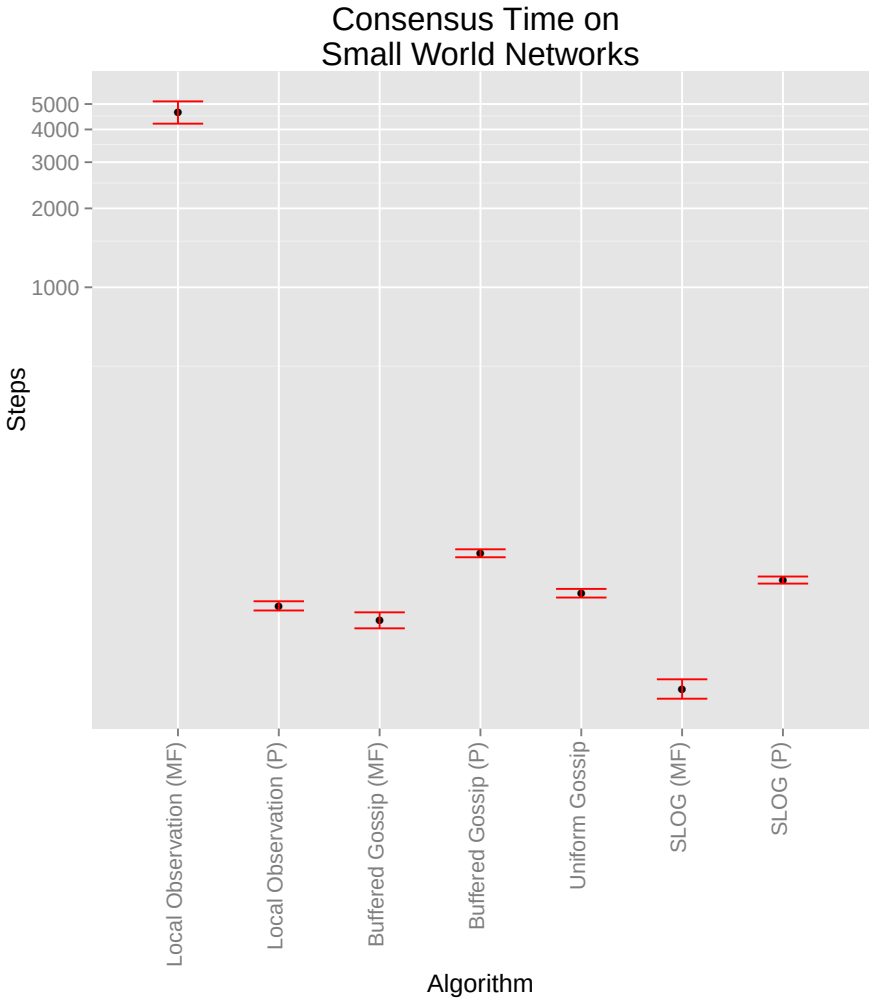

Figure 6.6: Data from the empirical comparison between information propagation algorithms, visualized as errorbar plots of the 95% confidence intervals for the mean consensus time on Newmann-Watts-Strogatz small world networks.

sip algorithm using maximum frequency selection ($\mu_{bg.max} = 52.49, \sigma_{bg.max} = 180.70$, 95% CI_μ $[48.76, 56.22]$), a local observation algorithm using proportional selection ($\mu_{lo.pro} = 59.82, \sigma_{lo.pro} = 114.96$, 95% CI_μ $[57.45, 62.20]$), and the uniform gossip algorithm ($\mu_{uni} = 67.07, \sigma_{uni} = 128.73$, 95% CI_μ $[64.41, 69.73]$).

Based on the observations for the experiments on Newman-Watts-Strogatz small world networks, as represented in figure 6.4 and figure 6.3, I conclude that, with a high probability,

- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is not greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a uniform gossip algorithm is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a uniform gossip algorithm is not greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a local observation algorithm using maximum frequency selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a local observation algorithm using maximum frequency selection is greater than the mean consensus time of a SLOG algorithm using proportional selection.

- The mean consensus time of a local observation algorithm using proportional selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a local observation algorithm using proportional selection is less than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a SLOG algorithm using maximum frequency selection is less than the mean consensus time of a SLOG algorithm using proportional selection.

In general, the observations represented in figure 6.6 and figure 6.5, suggest that a SLOG algorithm using maximum frequency selection is the best choice for fast consensus formation over an arbitrary Newmann-Watts-Strogatz small world network. These observations also suggest that the local observation algorithm using maximum frequency selection is the worst choice for consensus formation due to its lack of stability.

Lattice Networks

Figure 4.8 visualizes the experimental data from 300 randomly generated lattice networks using a standard box plot. The x-axis indicates the state update protocol used by each algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

In figure 6.7, one can observe that, unlike in the other networks, all seven algorithm appear to have fairly similar performance. The median consensus time of a local observation algorithm using maximum frequency selection is lowest, but this particular algorithm also has instances in which

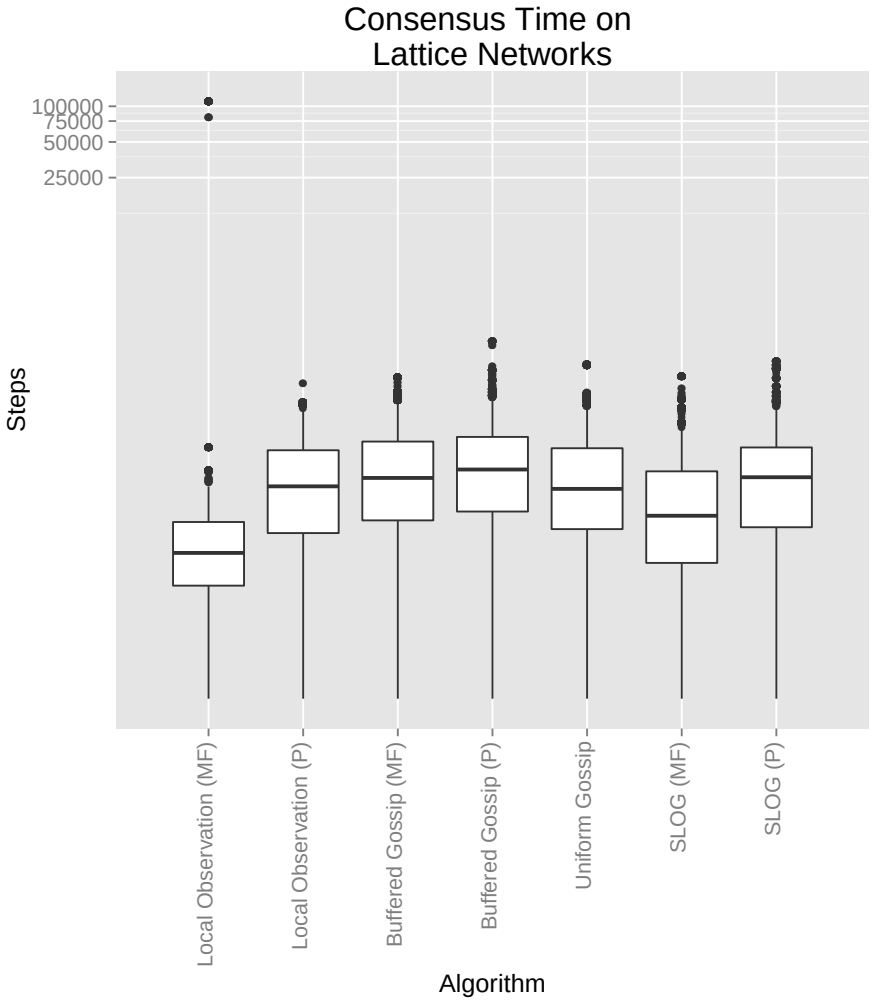

Figure 6.7: Data from the empirical comparison between information propagation algorithms, visualized as box plots that show the interquartile range, median value, and outliers of the consensus time on lattice networks.

consensus did not occur. A SLOG algorithm using maximum frequency selection has the second fastest median consensus time - and without any failures of convergence.

Figure 4.9 visualizes the mean consensus time of the lattice network data along with the 95% confidence interval of each mean. The x-axis indicates the state update protocol used by each

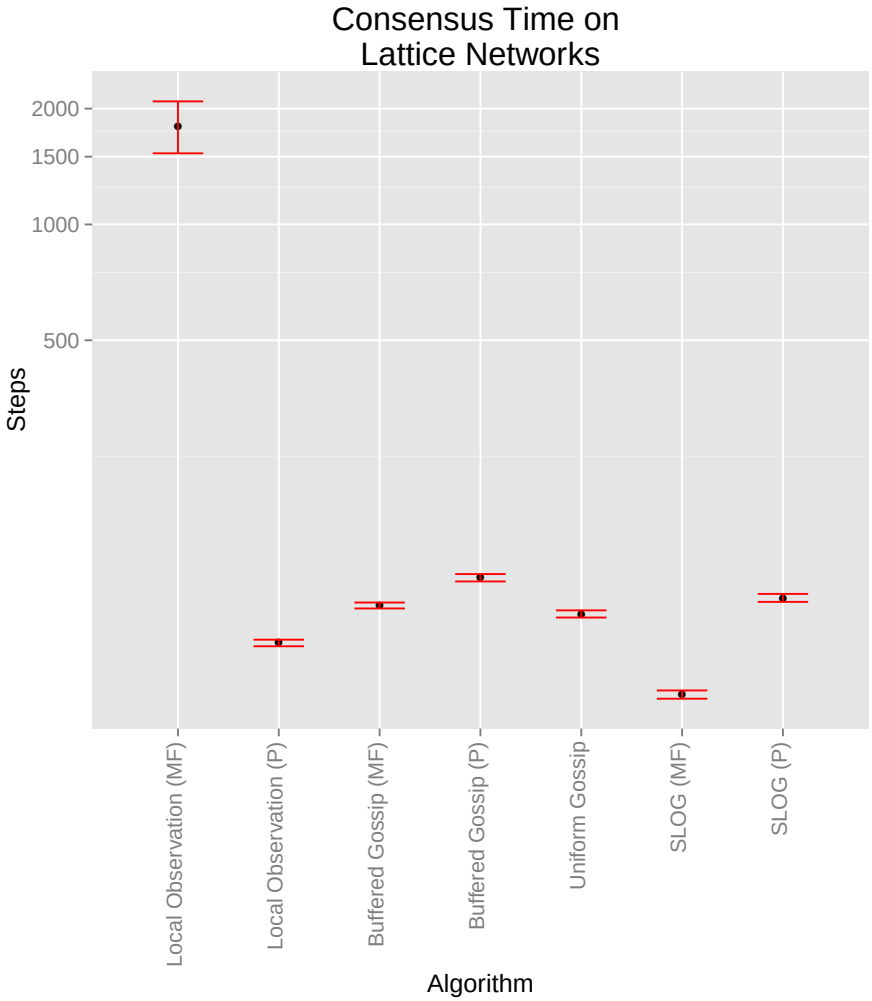

Figure 6.8: Data from the empirical comparison between information propagation algorithms, visualized as errorbar plots of the 95% confidence intervals for the mean consensus time on lattice networks.

algorithm. The y-axis indicates the number of steps until consensus is achieved. The y-axis has been transformed logarithmically in order to improve the overall visualization of the data; the data itself has not been transformed.

In figure 6.8, one can observe that the mean consensus time of a SLOG algorithm with maximum

frequency selection ($\mu_{slog.max} = 58.92, \sigma_{slog.max} = 71.93$ 95%, CI_μ $[57.43, 60.40]$) is less than all other algorithms studied in this experiment. The mean consensus time of a local observation algorithm using maximum frequency selection is the highest ($\mu_{lo.max} = 1800.76, \sigma_{lo.max} = 13833.05$ 95% CI_μ $[1514.93, 2086.59]$). The mean consensus time of a SLOG algorithm using proportional selection ($\mu_{slog.pro} = 105.89, \sigma_{slog.pro} = 123.04$, 95% CI_μ $[103.35, 108.43]$) is less than the mean consensus time of buffered gossip algorithms using proportional selection ($\mu_{bg.pro} = 119.69, \sigma_{bg.pro} = 128.80$ 95% CI_μ $[117.03, 122.35]$), but greater than the mean consensus time of a local observation algorithm using proportional selection ($\mu_{lo.pro} = 80.78, \sigma_{lo.pro} = 72.10$ 95% CI_μ $[79.29, 82.27]$), the uniform gossip algorithm ($\mu_{uni} = 96.12, \sigma_{uni} = 106.85$ 95% CI_μ $[93.91, 98.33]$), and a buffered gossip algorithm using maximum frequency selection ($\mu_{bg.max} = 101.26, \sigma_{bg.max} = 92.48$ 95% CI_μ $[99.35, 103.17]$).

Based on the observations for the experiments on lattice networks, as represented in figure 6.8 and figure 6.7, I conclude that, with a high probability,

- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using maximum frequency selection is not greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a buffered gossip algorithm using proportional selection is

greater than the mean consensus time of a SLOG algorithm using proportional selection.

- The mean consensus time of a uniform gossip algorithm is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a uniform gossip algorithm is not greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a local observation algorithm using maximum frequency selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a local observation algorithm using maximum frequency selection is greater than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a local observation algorithm using proportional selection is greater than the mean consensus time of a SLOG algorithm using maximum frequency selection.
- The mean consensus time of a local observation algorithm using proportional selection is less than the mean consensus time of a SLOG algorithm using proportional selection.
- The mean consensus time of a SLOG algorithm using maximum frequency selection is less than the mean consensus time of a SLOG algorithm using proportional selection.

In general, the observations represented in figure 6.8 and figure 6.7, suggest that a SLOG algorithm using maximum frequency selection is the best choice for fast consensus formation over an arbitrary Newmann-Watts-Strogatz small world network. These observations also suggest that the local observation algorithm using maximum frequency selection is the worst choice for consensus formation due to its lack of stability.

Impact of Network Topology

In regards to comparing the mean consensus time across network topology, the experimental results support the expectation that consensus time is sensitive to the topology of the agent communication network. The communication topology appears to produce a difference in the relative performance of information propagation algorithms using proportional selection or maximum frequency selection, as well as the uniform gossip algorithm. What is especially interesting, however, is that the mean and median consensus time resulting from the use of various algorithms appears to become similar on lattice-like networks.

Summary of Results

Consensus was not observed in all of our experiments. Local observation algorithms using maximum frequency selection failed to arrive at a consensus at least once on every type of network that we tested. All other algorithms, however, did reach consensus.

As with the results of my other experiments on involving buffered gossip algorithms and the uniform gossip algorithm, the topology of the communication network may be the most critical factor in the speed of consensus for SLOG algorithms. Evidence of this behavior is found in the observed consensus times across the four network topologies tested in these experiments – the mean consensus time varies more between the type of network than the state update protocol. Regardless of the network topology, however, a SLOG algorithm using maximum frequency selection always has both a lower median consensus time and a mean consensus time than a SLOG algorithm using proportional selection. There are instances in which a local observation algorithm using maximum frequency selection may perform better; but, that performance comes with the risk that consensus may not be achieved.

In regards to the relative performance improvement gained by combining push-based and pull-based transfer mechanisms into a stochastic transfer mechanism, that SLOG algorithms out-perform other information propagation algorithms with the same timing model, transfer protocol, and state update protocol. The SLOG algorithm using maximum frequency is guaranteed to produce a consensus, and produces that consensus faster than either local observation algorithms, buffered gossip algorithms, or the uniform gossip algorithm.

As a solution to the decentralized consensus problem, these results indicate that the SLOG algorithm may be a universally better solution than either buffered gossip algorithms or local observation algorithms using the same state update protocol.

CHAPTER 7: CONCLUSIONS

Having introduced three types of information propagation algorithm, I will now discuss the implications of my experimental result on the decentralized consensus problem in the discrete domain. Following this discussion, I will propose a number of avenues for future research. Finally, I will summarize the entirety of my research and present my conclusions.

Discussion

The data and observations from our three experiments on buffered gossip algorithms, local observation algorithms, and Stochastic Local Observation/Gossip (SLOG) algorithms suggests that SLOG algorithms using maximum frequency selection have the lowest mean consensus time of all algorithms examined. With the exception of local observation algorithms, the use of a proportional selection protocol always results in greater consensus times than an equivalent algorithm with a maximum frequency selection state update protocol. Local observation algorithms using a maximum frequency selection state update protocol are fast, but they are not always successful at producing a consensus. Buffered gossip algorithms using maximum frequency selection always reach a consensus, but not as fast as successful local observation algorithms. SLOG algorithms using maximum frequency selection combine local observation and buffered gossip to to produce an algorithm that always reaches a consensus, and does so nearly as fast as a successful local observation algorithm. Furthermore, the relative performance of the SLOG algorithm using maximum frequency selection was consistent across all examined network topologies. In every case, the SLOG algorithm using maximum frequency selection produced a consensus in the least amount of time relative to the other algorithms.

A SLOG algorithms using maximum frequency selection is most likely able to outperform its competition because of two primary factors: the use of a state update protocol that innately tries to build towards consensus, and the expected size of a node's buffer. The maximum frequency selection state update protocol is designed to select information based on a majority, and randomly if no majorities exist. Naturally, as nodes start to possess the same state value, local majorities will be reinforced and their value spread outwards. This behavior is in contrast to proportional selection, where it is possible for randomness to eat away at a majority. When maximum frequency selection is combined with a local observation transfer mechanism, a node can easily detect the presence of local majorities and so one would expect consensus to occur quickly. This does not always happen, however, because the consideration of every neighbor can lead to groups of nodes that get "stuck", as depicted figure 5. In a SLOG algorithm, a node considers the state of all of its neighbors with probability α. When $\alpha < 1$, there are times when a node considers only a subset of those neighbors. If only a subset of neighbors are considered, then a node is less likely to get stuck in a single state because it is possible for that node to consider two disjoint subsets of neighbors within two consecutive steps. This behavior, which results in the condition $E(|\beta_u|) < N(u)$, is at the core of the SLOG algorithm's stability.

With regards to the decentralized consensus problem and its areas of application, such as rendezvous, my research supports the use of a SLOG algorithm as the baseline solution, unless problem constraints prohibit their usage, or the problem structure allows the creation of specialized algorithms that can out-perform a SLOG algorithm. One such constraint that may occur is a limitation on the buffer size of a node to a value much smaller than the number of neighbors each node is expected to have. In this case, however, the SLOG algorithm could be modified to observe a subset of neighbors, instead of every neighbor; but, there is no guarantee that the performance of such an algorithm would be comparable. One example of beneficial problem structure would be the allowance of broadcast communication. If a node is able to broadcast its state value to every

neighbor, then it may be possible to obtain a consensus in less time than one would with a SLOG algorithm; but, this has not yet been tested.

Future Work

I have explored the consensus behavior of simple information propagation algorithms, but this exploration has been limited to questions concerning the possibility of consensus and the relative time in which consensus is reached among a specific set of information propagation algorithms.

There are at least four additional research questions that can build on the results presented in this dissertation.

E1: Can the average consensus time of an information propagation algorithm be estimated without simulation?

E2: What is this consensus behavior of additional state update protocols?

E3: How does error impact the speed of consensus?

E4: How do the algorithms introduced in this dissertation perform when implemented in real systems?

Question E1 is motived by a knowledge gap in this dissertation. I have successfully shown that a number of information propagation algorithms are guaranteed to produce a consensus, but I have said nothing as to how one might compute either the average consensus time or the bounds of the consensus time. My experimental results indicate that the bounds may be very wide, as some simulations finish in under 5 steps and others halt after tends of thousands of steps. With regards to calculating the average consensus time, however, absorbing Markov chains may prove to be a

good starting location. Preliminary research on this approach [22] shows that absorbing Markov chains can be used to model buffered gossip algorithms using proportional selection, but their construction quickly becomes infeasible as the size of the network exceeds 10-15 nodes. Another concern with computing the bounds on consensus time is related to scalability. Specifically, how does the consensus time scales with the size of the network or measures of centrality (e.g. average degree, betweenness or clustering coefficient)? This is a question of practicality, as exponential scaling would imply that even though consensus will eventually be reached on a large network, the time required to obtain it may be longer than the lifespan of the user. On the other hand, if the scaling is linear or logarithmic, then even large networks will form a consensus within a reasonable amount of time.

Question E2 is the most straightforward extension to this dissertation. I have focused only on two simple approaches to reducing the values in a buffer - select a value at random, or select the most represented value. Other possible approaches, of equal simplicity, might be to select the value least represented, or keep the current state value if the values in the buffer are not all equal to one another. It is also worth investigating more complicated reduction methods that incorporate the tracking of previous values, or wait multiple steps between state updates. The more state update protocols that are investigated, the better our understanding of consensus formation will become.

Question E3 is motivated by another knowledge gap in this dissertation. I have shown that the information propagation algorithms studied are robust to noise and node failure when it comes to consensus formation, but I do not fully explore how transmission error impacts the time required to research consensus. For instance, if a node using a local observation algorithm failed to observe all of of its neighbors, would that increase or decrease the mean consensus time? Likewise, what if a gossip algorithm mistakenly transmitted to more than one neighbor at a time? The answers to these questions could have profound implications to the use of these algorithms in the real world. Based on the differences in consensus behavior between a local observation algorithm using maximum

frequency selection and a SLOG algorithm using maximum frequency selection, I expect that the presence of error may be beneficial in certain circumstances. In the specific case of these two algorithms, the SLOG algorithm can be thought of as a local observation algorithm in which a node fails to observe all of its neighbors with probability $1 - \alpha$; the algorithm becomes stable in the presence of error.

Question E4 is motivated by practicality. This dissertation studies information propagation algorithms in a pure environment, where consideration is not given to hardware or software restrictions. In the real world, however, there are often constraints that must be be accounted for. These constraints have the potential to impact the consensus time of a system, although as long as the basic assumptions discussed in our proofs are met, consensus will still be possible. One example of such a real-world system might be a team of robots that must rendezvous at specific waypoints. In this example, there may be hardware limitations that restrict how often transmissions can be sent between the agents; or limitations with regards to how fast the CPU can process incoming data.

Conclusions

The task of the decentralized consensus problem for multi-agent systems is to design an algorithm that enables agents to communicate and exchange information such that, in finite time, every agent comes to possess the same information without the use of a centralized control mechanism.

The primary goal of this research is to introduce and provide supporting evidence for Stochastic Local Observation/Gossip (SLOG) algorithms as new solutions to the decentralized consensus problem for multi-agent systems that lack a centralized controller, with the additional constraints that agents act asynchronously, information is discrete, and all consensus options are equally preferable to all agents. Examples of where these constraints might apply include the spread of social norms

and conventions in artificial populations, rendezvous among a set of specific waypoints, and task allocation.

Prior to introducing SLOG algorithms, I derive the concepts of an information propagation process and an information propagation algorithm, and then use the structure provided by these concepts to define two algorithms that spread information across a network and solve the decentralized consensus problem: the buffered gossip algorithm and the local observation algorithm. The buffered gossip algorithm spreads information according to a push-based methodology and generalizes the well-known and widely-used uniform gossip algorithms. The local observation algorithm spreads information according to a pull-based methodology and generalizes multiple opinion dynamics models, including the voter model and the label propagation algorithm. I use linear algebra and probability to verify that these algorithms are solutions to the decentralized consensus problem.

SLOG algorithms are information propagation algorithms that combine the transmission mechanisms of buffered gossip algorithms and local observation algorithms into a single "hybrid" algorithm that is able to push and pull information within the local neighborhood. As with buffered gossip algorithms and local observation algorithms, I use linear algebra and probability to verify that these algorithms are solutions to the decentralized consensus problem.

I use a series of simulation experiments to study the performance of SLOG algorithms. These experiments compare the average speed of consensus formation between buffered gossip algorithms, local observation algorithms, and SLOG algorithms over four distinct network topologies. The uniform gossip algorithm is used as a baseline of measurement due its established history and presence in the related literature.

The data and observations from these experiments suggests that SLOG algorithms have the potential to solve the decentralized consensus problem faster than the experimental alternatives. These experiments also support the theoretical findings that SLOG algorithms will always reach a con-

sensus when a basic set of assumptions hold true: the network contains a directed spanning tree, and the nodes of that network employ both an asynchronous timing model and a selection-based state update protocol.

By leveraging models from the field of computational social science (opinion dynamics) and applying their core ideas to a standard problem in both control theory and computer science (consensus formation), I have been able to bridge a divide created by terminology and analytical technique and construct a novel solution to the decentralized consensus problem that is both fast and stable. This result has implications that reach beyond the context of the decentralized consensus problem by adding support to the idea that the hybridization of existing algorithms can open new avenues for advancing solutions to both existing and future problems.

LIST OF REFERENCES

[1] W. Ren, R. Beard, and E. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proceedings of the 2005 American Control Conference*, pp. 1859–1864, IEEE, 2005.

[2] W. Ren, R. Beard, and E. Atkins, "Information consensus in multivehicle cooperative control," *Control Systems, IEEE*, vol. 27, pp. 71–82, April 2007.

[3] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, pp. 215–233, Jan 2007.

[4] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, "Gossip Algorithms for Distributed Signal Processing," *Proceedings of the IEEE*, vol. 98, pp. 1847–1864, Nov. 2010.

[5] N. Alon, A. Barak, and U. Manber, "On disseminating information reliably without broadcasting.," in *ICDCS*, pp. 74–81, IEEE Computer Society, 1987.

[6] U. Feige, D. Peleg, P. Raghavan, and E. Upfal, "Randomized broadcast in networks," *Random Structures & Algorithms*, vol. 1, no. 4, pp. 447–460, 1990.

[7] J. Lin, A. Morse, and B. D. O. Anderson, "The multi-agent rendezvous problem," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, vol. 2, pp. 1508–1513 Vol.2, Dec 2003.

[8] J. Lin, A. Morse, and B. D. O. Anderson, "The multi-agent rendezvous problem - the asynchronous case," in *Proceedings of the 43nd IEEE Conference on Decision and Control*, vol. 2, pp. 1926–1931 Vol.2, Dec 2004.

[9] A. Pelc, "Disc 2011 invited lecture: Deterministic rendezvous in networks: Survey of models and results," in *Distributed Computing* (D. Peleg, ed.), vol. 6950 of *Lecture Notes in Computer Science*, pp. 1–15, Springer Berlin Heidelberg, 2011.

[10] J. Fang, A. Morse, and M. Cao, "Multi-agent rendezvousing with a finite set of candidate rendezvous points," in *Proceedings of the 2008 American Control Conference*, pp. 765–770, June 2008.

[11] C. D. Hollander and A. S. Wu, "Gossip-based solutions for discrete rendezvous in populations of communicating agents," *PLoS ONE*, vol. 9, November 2014.

[12] C. D. Hollander and A. S. Wu, "The current state of normative agent-based systems," *Journal of Artificial Societies and Social Simulation*, vol. 14, no. 2, 2011.

[13] C. D. Hollander and A. S. Wu, "Using the process of norm emergence to model consensus formation," in *Proceedings of the 5th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pp. 148 –157, October 2011.

[14] A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," *ACM SIGOPS Operating Systems Review*, vol. 22, no. 1, pp. 8–32, 1988.

[15] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman, "A survey of gossiping and broadcasting in communication networks," *Networks*, vol. 18, no. 4, pp. 319–349, 1988.

[16] J. Hromkovic, R. Klasing, B. Monien, and R. Peine, "Dissemination Of Information In Interconnection Networks (Broadcasting & Gossiping)," *Combinatorial Network Theory*, pp. 125–212, 1996.

[17] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pp. 482– 491, IEEE, 2003.

[18] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[19] A.-M. KerMarchrec and M. van Steen, "Gossiping in distributed systems," *SIGOPS Operating Systems Review*, vol. 41, no. 5, pp. 2–7, 2007.

[20] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 565–574, IEEE, 2000.

[21] D. Kempe, J. Kleinberg, and A. Demers, "Spatial gossip and resource location protocols," *Journal of the ACM*, vol. 51, pp. 943–967, Nov. 2004.

[22] C. D. Hollander and A. S. Wu, "Distributed consensus formation through unconstrained gossiping," *Computer Research Repository (CoRR)*, vol. abs/1301.2722, 2013.

[23] C. Castellano, S. Fortunato, and V. Loreto, "Statistical physics of social dynamics," *Reviews of Modern Physics*, vol. 81, p. 591, May 2009.

[24] P. Krapivsky and S. Redner, "Dynamics of majority rule in Two-State interacting spin systems," *Physical Review Letters*, vol. 90, June 2003.

[25] M. Mobilia and S. Redner, "Majority versus minority dynamics: Phase transition in an interacting two-state spin system," *Phys. Rev. E*, vol. 68, p. 046106, Oct 2003.

[26] T. M. Liggett, *Interacting Particle Systems*. Springer Berlin Heidelberg, 2005.

[27] V. Sood and S. Redner, "Voter model on heterogeneous graphs," *Physical Review Letters*, vol. 94, p. 178701, May 2005.

[28] F. Schweitzer and L. Behera, "Nonlinear voter models: the transition from invasion to coexistence," *The European Physical Journal B*, vol. 67, no. 3, pp. 301–318, 2009.

[29] M. Yildiz, R. Pagliari, A. Ozdaglar, and A. Scaglione, "Voting models in random networks," in *Information Theory and Applications Workshop (ITA), 2010*, pp. 1–7, Jan 2010.

[30] U. N. Raghavan, R. Albert, and S. KuMarcha, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E*, vol. 76, p. 036106, Sep 2007.

[31] D. Mosk-Aoyama and D. Shah, "Computing separable functions via gossip," in *Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing*, PODC '06, (New York, NY, USA), pp. 113–122, ACM, 2006.

[32] L. Lamport, "Paxos made simple," *ACM SIGACT News*, vol. 32, no. 4, pp. 18–25, 2001.

[33] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems*, vol. 16, pp. 133–169, May 1998.

[34] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, (Berkeley, CA, USA), pp. 305–320, USENIX Association, 2014.

[35] D. Kempe and J. Kleinberg, "Protocols and impossibility results for gossip-based communication mechanisms," in *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 471–480, IEEE Comput. Soc, 2002.

[36] A. J. Ganesh, A. M. KerMarchrec, and L. Massoulie, "Peer-to-peer membership management for gossip-based protocols," *IEEE Transactions on Computers*, vol. 52, no. 2, pp. 139– 149, 2003.

[37] B. Pittel, "On Spreading a Rumor," *SIAM Journal on Applied Mathematics*, vol. 47, pp. 213–223, Sept. 1987.

[38] P. T. Eugster, R. Guerraoui, A.-M. M. KerMarchrec, and L. Massoulie, "Epidemic information dissemination in distributed systems," *Computer*, vol. 37, pp. 60–67, May 2004.

[39] A. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic gossip: efficient aggregation for sensor networks," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, pp. 69–76, IEEE, 2006.

[40] A. D. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic Gossip: Efficient Averaging for Sensor Networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 3, pp. 1205–1216, 2008.

[41] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2748–2761, 2009.

[42] A. Kashyap, T. Basar, and R. Srikant, "Consensus with Quantized Information Updates," in *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 2728–2733, IEEE, Dec. 2006.

[43] T. D. Chandra, R. Griesemer, and J. Redstone, "Paxos made live: An engineering perspective," in *Proceedings of the 26th Annual ACM Symposium on Principles of Distributed Computing*, PODC '07, (New York, NY, USA), pp. 398–407, ACM, 2007.

[44] A. Mocanu and C. Bădică, "Bringing paxos consensus in multi-agent systems," in *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*, WIMS '14, (New York, NY, USA), pp. 51:1–51:6, ACM, 2014.

[45] A. Campbell and A. S. Wu, "On the significance of synchroneity in emergent systems," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Sys-*

tems - Volume 1, AAMAS '09, (Richland, SC), pp. 449–456, International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[46] J. P. Hecker, A. S. Wu, J. A. Herweg, and J. C. Sciortino, Jr., "Team-based resource allocation using a decentralized social decision-making paradigm," vol. 6964, pp. 696409–696409–9, SPIE, 2008.

[47] X. Liu and T. Murata, "How Does Label Propagation Algorithm Work in Bipartite Networks?," in *Proceedings of the IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09.*, vol. 3, pp. 5–8, 2009.

[48] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, (Pasadena, CA USA), pp. 11–15, Aug. 2008.

[49] T. E. Oliphant, "Python for scientific computing," *Computing in Science & Engineering*, vol. 9, no. 3, 2007.

[50] M. E. J. Newman and D. J. Watts, "Renormalization group analysis of the small-world network model," *Physics Letters A*, vol. 263, pp. 341–346, 1999.

[51] B. L. WELCH, "The generalization of student's problem when several different population varlances are involved," *Biometrika*, vol. 34, no. 1-2, pp. 28–35, 1947.

[52] M. A. Nowak and R. M. May, "Evolutionary games and spatial chaos," *Nature*, vol. 359, p. 826, 1992.

[53] B. A. Huberman and N. S. Glance, "Evolutionary games and computer simulations," *Proceedings of the National Academic of Sciences of the United States of America*, vol. 90, no. 16, pp. 7716–7718, 1993.