A Study on Loosely-Stabilizing Algorithms for Maximal Independent Set with Unreliable Communications

Submitted to

Graduate School of Information Science and Technology

Osaka University

July 2023

Rongcheng DONG

List of Related Publications

Journal Papers

1. Rongcheng Dong, Yuichi Sudo, Taisuke Izumi, and Toshimitsu Masuzawa, "Loosely-Stabilizing Maximal Independent Set Algorithms with Unreliable Communications," *Theoretical Computer Science*, Elsevier, Vol. 937, p.69–84, 2022.

2. Rongcheng Dong, Taisuke Izumi, Naoki Kitamura, Yuichi Sudo, and Toshimitsu Masuzawa, "Loosely-Stabilizing Algorithms on Almost Maximal Independent Set," *IEICE Transactions on Information and Systems*, 2023 (to appear).

Conference Paper

3. Rongcheng Dong, Yuichi Sudo, Taisuke Izumi, and Toshimitsu Masuzawa, "Loosely-Stabilizing Maximal Independent Set Algorithms with Unreliable Communications," *Proceedings of the 23rd International Symposium on Stabilization, Safety, and Security of Distributed Systems*, LNCS 13046, p.335–349, Virtual Event, Nov. 2021.

Abstract

A distributed system comprises a set of autonomous processes, such as computers and processors, and communication links that connect processes in the system. Numerous processes and links in the distributed system make it more flexible and scalable compared with the traditional centralized system. Therefore, the distributed system has obtained more and more attention from both theoretical research and industry. It has extensive applications and is widely used all over the world, for example, the Internet, wireless sensor networks, cloud computing, multi-processor computers, and distributed databases. The *maximal independent set* (MIS) problem is one of the most fundamental problems in the field of distributed computing. Many problems of distributed computing can be reduced to the MIS problem or use MIS as a subroutine, including the matching problem and the set cover problem. Moreover, the MIS problem is applied in numerous areas, such as software engineering, economics, and biology.

However, the complexity of the distributed system also brings instability, since the crashes and failures of processes and communication links are always non-negligible. Self-stabilization, which is originally proposed by Dijkstra in 1974, is a promising paradigm for designing distributed systems that can autonomously adapt to dynamics caused by transient faults and topology changes of networks. A self-stabilizing system is characterized by two properties called *convergence* and *closure*. The convergence allows the system to eventually reach legitimate configurations (i.e., configurations satisfying the problem specification) regardless of the initial configuration, and the closure makes the system stay in legitimate configurations *forever*. This dissertation focuses on self-stabilizing algorithms in the distributed system with unreliable communications between processes in the graph, i.e., each communication channel suffers the corruption of the transmitted messages (stochastically or adversarially). An inherent limitation of conventional self-stabilizing

algorithms is that it requires the system to be fault-free during its convergence. Thus, the design of self-stabilizing algorithms under the threat of perpetual faults is often recognized as a challenging problem. Probabilistic error models, where message corruption is modeled as a stochastic event that the adversary cannot control, are widely accepted as reasonable assumptions of perpetual faults. It is not only standard in information theory but also popular in distributed computing. Self-stabilizing solutions are, however, ruled out for most non-trivial problems. More precisely, it allows an execution starting from any legitimate configuration to some non-legitimate one with a non-zero probability.

To circumvent the impossibility of self-stabilization in probabilistic error models, this dissertation focuses on *loose-stabilization*, which is a relaxed variant of conventional self-stabilization. While keeping the same convergence property with self-stabilization, loose-stabilization relaxes the closure property: the system is allowed to deviate from legitimate configurations after being legitimate for a long time in expectation with high probability. Loose-stabilization is practically equivalent to self-stabilization if the duration when the system stays in legitimate configurations (called *holding time*) is much longer (e.g., exponentially longer) than the time required to reach a legitimate configuration (called *convergence time*). Therefore, even if some unexpected error occurs and causes the system to become illegitimate, in a relatively short time the system can converge to legitimate configurations again and keeps being legitimate for a sufficiently long time with high probability.

In this dissertation, we consider loose-stabilizing algorithms for the MIS problem adapting to unreliable communication links between processes. Chapter 2 is devoted to the formal definitions and concepts used in the dissertation, including the distributed system, the computational model, the erroneous communication model, the MIS problem, and loose-stabilization. In Chapter 3, we present three systematic approaches for designing loose-stabilizing algorithms in probabilistic error models: the *redundant-state*, the *step-up*, and the *repetition* approaches. All the approaches assume the atomic-state model (where each process can atomically change its state depending on its state and the neighbors' current states) with the stochastic scheduler and follow certain kinds of error-correction/detection mechanisms. The basic ideas of error-correction/detection mechanisms are straightforward and adopted from information theory and coding theory. In Chapter 4, we present a relaxed notion of the MIS problem, named *almost MIS* (ALMIS), and

show that the loosely-stabilizing algorithm for the MIS problem in Chapter 3 can be adopted to overcome the trade-off between space and holding time when considering ALMIS.

Contents

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background

A distributed system consists of a collection of independent processes, such as computers and processors, interconnected by communication links. The presence of multiple processes and links within the distributed system enhances its flexibility and scalability when compared to traditional centralized systems. As a result, the distributed system has gained significant attention from both theoretical research and industry. It finds extensive applications and is widely utilized worldwide, encompassing areas such as the Internet, wireless sensor networks, cloud computing, multi-processor computers, and distributed databases. Among the fundamental problems in the field of distributed computing, the *maximal independent set* (MIS) problem holds significant importance. An MIS is sometimes used to realize the local mutual exclusion, which avoids the collision of actions between neighboring processes. Such collision avoidance plays an important role to guarantee correct behaviors in real distributed systems. Distributed algorithms for MIS are used as subroutines for many distributed graph problems, such as the node coloring problem, the matching problem and the set cover problem.

However, the inherent complexity of distributed systems also introduces instability due to the presence of process crashes and communication link failures, which cannot be disregarded. Self-stabilization, initially proposed by Dijkstra in 1974 [1], offers a promising paradigm for designing distributed systems capable of autonomously adapting to dynamic changes caused by

transient faults and network topology alterations. A self-stabilizing system is characterized by two essential properties known as *convergence* and *closure*. Convergence ensures that the system eventually reaches legitimate configurations, satisfying the problem specification, regardless of the initial configuration. On the other hand, closure guarantees that the system remains in legitimate configurations *indefinitely*. This dissertation focuses on self-stabilizing algorithms within distributed systems that experience unreliable communications between processes in the graph. In this context, each communication channel suffers from message corruption, either stochastically or due to adversarial factors.

The conventional self-stabilizing algorithm possesses an inherent limitation, requiring the system to be fault-free during the whole execution of the algorithm. Consequently, designing self-stabilizing algorithms in the presence of perpetual faults is widely recognized as a challenging problem. Probabilistic error models, which assume that message corruption is a stochastic event beyond the adversary's control, are commonly accepted as reasonable representations of perpetual faults. These models are not only standard in information theory but also popular in distributed computing. Nonetheless, such error models typically render self-stabilizing solutions infeasible for most non-trivial problems. Specifically, they allow for the occurrence of executions starting from any legitimate configuration that leads to an illegitimate configuration with a non-negligible probability.

To overcome the inherent impossibility of self-stabilization under probabilistic error models, this dissertation adapts the concept of *loose-stabilization*, which represents a relaxed variant of conventional self-stabilization. Unlike other weakened forms of self-stabilization that relax the convergence property, such as probabilistic self-stabilization, pseudo-stabilization, and weak-stabilization, loose-stabilization does not impose any constraints on initial configurations or convergence periods, and it is the first trial to relax the closure property. While retaining the convergence property of self-stabilization, loose-stabilization permits the system to deviate from legitimate configurations after remaining legitimate for a significant period, on average, with a high probability. Due to the loosed-closure property, loose-stabilization can recover from not only transient faults, as self-stabilization and its other variants do, but also intermittent failures that more commonly occur in real systems.

Loose-stabilization is practically equivalent to self-stabilization when the duration during

which the system remains in legitimate configurations (referred to as the *holding time*) is considerably longer (e.g., exponentially longer) than the time required to achieve a legitimate configuration (known as the *convergence time*) from an arbitrary initial configuration. Consequently, even if an unexpected error occurs, causing the system to become illegitimate, it can relatively quickly converge back to legitimate configurations and remain legitimate for a sufficiently long period, with a high probability.

In terms of overhead, self-stabilizing and loose-stabilizing algorithms exhibit differences compared to non-fault-tolerant algorithms. They may incur additional execution time, memory requirements, and increased information exchange. However, it is important to note that for many problems, self-stabilization and loose-stabilization are considered lightweight fault tolerance techniques in comparison to classical robust approaches [2]. This is primarily due to the *optimistic* (also called *non-masking*) approach adopted by self-stabilization and loose-stabilization, as opposed to the *pessimistic* (also called *masking*) approach of robust algorithms [3]. Pessimistic approaches aim to prevent deviations from the system's specification, particularly in the presence of faults, by extensively validating each step. In contrast, the optimistic approach does not anticipate unexpected faults, which may result in temporary inconsistencies. Nevertheless, it guarantees that any deviation from the specification is only temporary or has a duration that is negligible compared to the time spent in a satisfactory state. Despite occasional inconsistencies, the optimistic nature of self-stabilization and loose-stabilization allows them to be more efficient and lightweight compared to robust approaches. The trade-off lies in their prioritization of resilience and rapid fault recovery over strict prevention of deviations.

1.2 Overview of This Dissertation

In this dissertation, we consider loose-stabilizing algorithms for the MIS problem in the arbitrary topology adapting to unreliable communications, that is, the communication links between processes in the network are under probabilistic crashes or failures so that processes erroneously communicate with each other.

Chapter 2 is devoted to the formal definitions and concepts used in the dissertation, including the distributed system, the computational model, the erroneous communication model, the MIS

problem, and loose-stabilization.

In Chapter 3, we consider two different error models for erroneous communications: the uniformly distributed error model and the arbitrarily distributed error model. We propose three systematic approaches in the network with erroneous communications to achieve loose-stabilization with the relaxed closure property and apply these approaches to design loosely-stabilizing algorithms for the MIS problem with different assumptions and performances: the *redundant-state*, the *step-up*, and the *repetition* approaches. All the approaches assume the atomic-state model (where each process can atomically change its state depending on its state and the neighbors' current states) with the stochastic scheduler and follow certain kinds of error-correction/detection mechanisms. The basic ideas of error-correction/detection mechanisms are straightforward and adopted from information theory and coding theory.

The three algorithms have different assumptions and performances, which are summarized in Table 3.1. The redundant-state algorithm requires a constant maximum degree in the network, and assumes the uniformly distributed error model. Its maximum expected convergence time is $O(\log n)$, minimum expected holding time is $poly(n)$, and space complexity per node is $O(\log n)$, where n is the size of the network. The step-up algorithm only assumes the uniformly distributed error model, with the maximum expected convergence time of $O(n \log^3 n)$, the minimum expected holding time of $poly(n)$, and the space complexity of $O(\log n)$. The repetition algorithm requires a unique identifier for each process and assumes the arbitrarily distributed error model, with the maximum expected convergence time of $O(n^2 \log n)$, the minimum expected holding time of $e^{\Omega(n)}$, and the space complexity of $O(n\Delta \log n)$, where Δ is the maximum degree in the network.

A major limitation of the above three algorithms is that, we have to make trade-offs between the performance and space complexity: we can make the holding time much longer than the convergence time only if processes in the network have a larger capacity. On the other hand, we have the following observation on many of loosely-stabilizing algorithms including the three algorithms mentioned above: an execution period of holding legitimate configurations terminates with a *small* violation of the legitimacy of configurations. In the case of MIS, for example, it is typically terminated by the event that a small number of processes violate the specification of MIS. Conversely, even if the system drops out from legitimate configurations (after a holding period), almost all processes still locally satisfy the specification of MIS. Then one can also expect

that the system quickly recovers a legitimate configuration again. Hence the system is expected to keep a large portion of processes "locally" legitimate, for a period much longer than the expected holding time with respect to the specification of MIS. To capture such a behavior, in Chapter 4, we newly propose and formulate a relaxed notion of the MIS problem, namely *almost MIS* (ALMIS). Our formulation divides the specification of MIS into two properties referred to as *independence* and *maximality*, and quantify the magnitude of violation for each property. The precise definition of ALMIS follows a parametric specification with an acceptable level of violation.

After formulating the definition of ALMIS, we also demonstrate the loosely-stabilizing redundant-state algorithm presented in Chapter 3 can achieve the exponential minimum expected holding time regarding ALMIS with a reasonably small acceptable level of violation, while it still keeps $O(\log n)$ maximum expected convergence time (refer to Table 4.1). In addition, one can also show that the algorithm keeps the precise MIS within most of the holding periods. By adjusting the value of the parameter k_{AL} that represents the extent that the requirements of MIS are relaxed, we can make a trade-off between the quality of ALMIS and the performance of the algorithm. When $k_{AL} = 0$, the ALMIS problem degenerates to the MIS problem.

1.3 Related Works

1.3.1 Maximal Independent Set Problem

The MIS problem is one of the most fundamental problems in the study of distributed graph algorithms. Luby [4] proposed a classic randomized distributed MIS algorithm with round complexity $O(\log n)$ based on Monte Carlo algorithms. It was first improved by Barenboim et al. [5], who proposed an MIS algorithm running in $O(\log^2 \Delta + 2^{O(\sqrt{\log \log n})})$ rounds where Δ is the maximum degree, while the message size is up to $poly(\Delta \log n)$ bits. Ghaffari [6] proposed a highly straightforward randomized MIS algorithm focusing on the local complexity. By modifying this algorithm, he achieved the current state-of-the-art global complexity of $O(\log \Delta + 2^{O(\sqrt{\log \log n})})$ rounds.

There had been a significant gap, which is nearly exponential, between randomized and deterministic MIS algorithms until a polylogarithmic-time deterministic MIS algorithm was provided by Rozhon et al. [7] based on network decomposition. Based on a variant of the

network decomposition in[7], the time complexity was improved to $O(\log^5 n)$ rounds in [8]. Currently, the known fastest deterministic MIS algorithm was proposed by Faour et al. [9], which has a time complexity of $O(\log^2 \Delta \cdot \log n)$ rounds. Their algorithm is also the first direct polylogarithmic deterministic MIS algorithm that is not based on network decomposition.

The weak variant of MIS has been proposed in several works. In [10], authors proposed the maximal nearly independent set, which relaxed the first requirement of MIS such that the number of chosen processes that are adjacent to each other is not too large, and based on this they proposed parallel greedy approximation algorithms for set cover problems. On the other hand, the second requirement of MIS, i.e., the maximality requirement, is weakened in [5] and [11], such that there exists a restrictive number of processes that are not independent nor adjacent to any independent process in the network. Such an almost-maximal independent set is used as an intermediate result to compute a strict MIS.

There are various applications of the MIS problem in different domains. Many other distributed graph algorithms use MIS as a subroutine. In [4] and [12], it was shown that maximal matching, vertex coloring, and edge coloring problems can be reduced to the MIS problem. MIS is also applied in the design of wireless sensor networks [13, 14, 15], resource allocation [16, 17, 18], coding theory [19, 20], and biological systems [21, 22, 23].

1.3.2 Self-Stabilization

Self-stabilization [1] is a promising paradigm for designing distributed systems that can autonomously adapt to dynamics caused by transient faults and topology changes of networks. Hedetniemi et al. [24] proposed a simple self-stabilizing MIS algorithm with constant round complexity while assuming a centralized scheduler. Arapoglu et al. [15] proposed a self-stabilizing MIS algorithm under the fully distributed scheduler running in $\max\{3n-6, 2n-1\}$ moves (the total number of process actions). Turau [25] considered randomization and proposed a randomized self-stabilizing MIS algorithm in the synchronous model that converges in $O(\log n)$ rounds. Afek et al. [26] proposed a self-stabilizing alternating bit protocol while considering message loss. Dolev et al. [27] proposed a self-stabilizing data-link protocol that emulates a reliable FIFO communication channel over unreliable non-FIFO channels, where messages could be lost, duplicated, created or reordered.

Although self-stabilization is an elegant and powerful tool to achieve fault tolerance, due to the two strict requirements of self-stabilization, convergence, and closure, sometimes it is difficult or even impossible to design one. Therefore, many researchers are devoted to finding the relaxed versions of self-stabilization, relaxing either on the convergence property or the closure property or both. Israeli et al. [28] relaxed the convergence property and proposed two probabilistic self-stabilizing mutual exclusion algorithms that can achieve the convergence property with probability one. Gouda [29] proposed weak stabilization and showed that if the system has a finite number of configurations and its execution is strongly fair, the weak stabilization can imply self-stabilization. Lin et al. [30] also relaxed the convergence property by proposing the notion of quasi-self-stabilization, where the algorithm can act as a self-stabilizing one if in the initial configuration all program pointers are set to zero, and applied it to mutual exclusion algorithms. Devismes et al. [31] investigated the weak and probabilistic stabilization, and proved that a deterministic weakly stabilizing algorithm can be transformed into a probabilistic stabilizing algorithm. The k-stabilization [32, 33, 34] solves problems that have no deterministic self-stabilizing solutions, such as the leader recovery problem in anonymous networks, by requiring that initial configurations can only be the result of at most k faults.

1.3.3 Loose-Stabilization

The notion of (α, β)-loose-stabilization, where α and β are convergence and holding time respectively, was first proposed [35] to circumvent the impossibility that the self-stabilizing leader election problem cannot be solved in the population protocol model unless the exact number of agents is available to agents in advance [36]. In [35], authors proposed a $(O(nN \log n), \Omega(Ne^N))$-loosely-stabilizing leader election algorithm in a complete graph using the population protocol model, where N is the upper bound of n. In [37], Sudo et al. generalized the topology to arbitrary graphs while requiring identifiers or random numbers. Sooner, they showed that such requirements could be eliminated and proposed another algorithm in [38]. Izumi solved the same problem and optimized the convergence time to linear while keeping the holding time to be exponential in [39]. In [40], Sudo et al. further improved the convergence time to be polylogarithmic, while the holding time is no longer exponential but polynomial with an arbitrarily large degree. In [41], Sudo et al. proposed a time-optimal loosely-stabilizing leader election protocol with logarithm

convergence and polynomial holding times in expectation. Feldmann et al. [42] first applied loose-stabilization to the message-passing model on server-client networks. They proposed a $(O(polylog(c(p_{min}^{-1} + n^3))), \Omega(n^c))$-loosely-stabilizing congestion control algorithm, where c is a parameter that can be chosen depending on the context, n is the number of clients and p_{min} is the minimum probability that a client will send a message to the server.

1.3.4 Erroneous Communications

The assumption of erroneous communication channels between processes in a network is more realistic and desirable when designing distributed algorithms. Afek et al. [26] presented a self-stabilizing alternating bit protocol between a transmitter and a receiver with an unreliable communication medium, where an oracle decides the occurrence of transient errors. They achieved self-stabilization by using a sequence number with a larger size than ordinary data-link protocols. In [43], authors analyzed the binary majority consensus problem with the presence of probabilistic message loss in a network where all processes (called agents in [43]) can communicate with all the others. In their approach, each process sends its state to all the other processes and decides its own state by choosing the majority of the results received from other processes for the next round, and within three rounds, processes reach a consensus with probability 1. Shirvanimoghaddam et al. [44] considered the federated learning problem with an erroneous communication model where the links between devices and the central node erase the packet with a parameterized probability. By designing the mechanism that the central node uses the past local information if the new packet is lost in the communication link from a device, the authors proved that the federated learning algorithm converges to the same global parameter as the algorithm converges to without any communication error.

1.4 Organization of This Dissertation

This dissertation consists of five chapters. In Chapter 2, we describe definitions of our computational model, probabilistic erroneous communication model, the MIS problem, and loose-stabilization. In Chapter 3, we propose three algorithms to solve the MIS problem with erroneous communications. In Chapter 4, we propose the definition of (f, g)-ALMIS and show that the

algorithm proposed in Chapter 3 can overcome the trade-off between space and holding time regarding ALMIS. We conclude this dissertation in Chapter 5.

Chapter 2

Preliminary

In this chapter, we describe definitions of a distributed system, the computational model, unreliable communication models, the MIS problem, and the notion of loose-stabilization.

Distributed System. A distributed system comprises a set of autonomous processes and the communication links that connect the processes. We abstract such a system as an undirected graph $G = (V, E)$: the vertex set V represents the set of n processes where $n \geq 1$, and the edge set $E \subseteq \{\{p, q\} \mid p, q \in V, p \neq q\}$ represents the set of communication links. If $\{p, q\} \in E$, we say p and q are neighbors of each other and can communicate with each other.

The set of neighbors of a process p is denoted by $N(p) = \{p \mid \{p, q\} \in E\}$, and p can distinguish each of its neighbors by some local labeling mechanism. Denote the degree of process p by $\Delta_p = |N(p)|$ and the maximum degree among all processes by $\Delta = \max\{\Delta_p \mid p \in V\}$. In this dissertation, we have no assumptions on the topology of G. In other words, G has an arbitrary topology.

Denote $N^*(I)$ as the set of the processes that are not in the set $I \subseteq V$ and adjacent to the processes in I, i.e., $N^*(I) = \{p \in V \setminus I \mid N(p) \cap I \neq \emptyset\}$. For any $I \subseteq V$, we denote the *edge boundary* of I by $\partial(I) = \{\{p, q\} \in E \mid p \in I, q \notin I\}$.

Computational Model. The computational model used in this dissertation is the atomic-state model: each process p has local variables $v_1, v_2, \ldots$, where the number and the domain of variables are decided by algorithms, and the values of variables of p define the state of p. A configuration of G is defined by the states of n processes in G. Each process can read its state and all of its

neighbors' states but can update only its state.

Synchrony or asynchrony of a system is characterized by the *scheduler*, which decides the set of activated processes at each time step. We assume the uniformly distributed scheduler U in this dissertation: at each *time step* (*step* for short hereafter), each process is independently activated with a constant probability ϕ. Note that the analysis and result presented in Chapters 3 and 4 hold even if $\phi = 1$ (so-called the synchronous scheduler), since ϕ is only a constant factor. The activated process reads the states of all its neighbors and its own, then does some local computations, and updates its state if necessary. We omit the notation of the scheduler U in the following for simplicity. In addition, since any time measurement depends on ϕ, and the asymptotic bound is the same as far as ϕ is a constant, we do not explicitly describe such a dependency.

In this dissertation, we say that an event occurs with high probability in the system if it occurs with probability more than $1 - 1/n^a$ for some constant $a > 0$ over all possible executions.

Erroneous Communication Models. We assume that each time a process p tries to read the value of a variable var_q from process $q \in N(p)$, process p obtains an incorrect result with a constant probability ρ. We introduce two error models with respect to unreliable communication: the *uniformly distributed error* model (denoted as M_u) and the *arbitrarily distributed error* model (denoted as M_a). In the *uniformly distributed error* model, when the erroneous communication occurs, p gets v as the current value of var_q with probability $1/(|Var_q|-1)$ for each $v \in Var_q \backslash \{v_q\}$, where Var_q is the set of all variable-values of var_q and v_q is the true value of var_q. In this model, we assume $0 < \rho < 1 - \epsilon$ for some positive constant ϵ. In the *arbitrarily distributed error* model, when the erroneous communication occurs, the resultant value p reads from q is arbitrary (i.e., chosen by the adversary). In this model, we assume $0 < \rho < 1/2 - \epsilon$.

Given an initial configuration C_0, the scheduler U, and the error model M, we define $E_{\mathcal{A}}(C_0, M)$ as the set of all possible executions of algorithm $\mathcal{A}$, where each execution is an infinite sequence of configurations $C_0, C_1, \cdots, C_i, C_{i+1}, \cdots$, where C_{i+1} is obtained by taking a step from C_i for all $i \geq 0$ under the error model M. Each execution has the probability of its occurrence, which is determined by U, M, and $\mathcal{A}$. A specification SP is a predicate over a configuration, which is the formal definition of the problem to be solved.

MIS Problem. A set of processes $I \subseteq V$ is called an *independent set* if $\{p, q\} \notin E$ holds

for any $p, q \in I$. In addition, if I is not a proper subset of any other independent set, then I is a *maximal* independent set (MIS).

Loose-Stabilization. Denote $\mathcal{C}$ as the set of all possible configurations of a distributed system G. Given an algorithm $\mathcal{A}$, for any $\mathcal{S} \subseteq \mathcal{C}$ and $C \in \mathcal{C}$, define $ECT_{\mathcal{A}}(C, \mathcal{S}, M)$ as the expected number of steps until executions in $E_{\mathcal{A}}(C, M)$ reach a configuration in $\mathcal{S}$ under the scheduler U and the error model M, where the notion ECT stands for the expected convergence time. For any configuration $C \in \mathcal{C}$ and a problem $\mathcal{P}$, we define $EHT_{\mathcal{A}}(C, SP_{\mathcal{P}}, M)$ as the expected number of steps until the executions in $E_{\mathcal{A}}(C, M)$ deviate from the specification $SP_{\mathcal{P}}$ of $\mathcal{P}$ for the first time under the scheduler U and the error model M, where the notion EHT stands for the expected holding time.

Definition 2.0.1. *An algorithm $\mathcal{A}$ under the scheduler U and the error model M is an (α, β)-loosely-stabilizing algorithm for problem $\mathcal{P}$ with specification $SP_{\mathcal{P}}$ if there exists a set $\mathcal{S}$ of configurations satisfying:*

- $\max_{C \in \mathcal{C}} ECT_{\mathcal{A}}(C, \mathcal{S}, M) \leq \alpha$
- $\min_{C \in \mathcal{S}} EHT_{\mathcal{A}}(C, SP_{\mathcal{P}}, M) \geq \beta$

We call α and β the maximum expected convergence time and the minimum expected holding time of $\mathcal{A}$, respectively. A configuration in $\mathcal{S}$ is called a safe configuration. Intuitively, loose-stabilization requires that an execution starting from any configuration reaches a safe configuration in a short time (α in Definition 2.0.1), and after that, the execution satisfies the problem specification for a sufficiently long time (β in Definition 2.0.1).

Chapter 3

Loosely-Stabilizing MIS Algorithms with Unreliable Communications

3.1 Introduction

3.1.1 Background

An inherent limitation of conventional self-stabilizing algorithms is that it requires the system to be fault-free during its convergence. Thus, the design of self-stabilizing algorithms under the threat of perpetual faults is often recognized as a challenging problem. The adversarial message corruption is one of the popular and strong models of perpetual faults, where at each time step the adversary chooses a set of links (whose size is typically constrained) and modifies the messages transferred through the chosen edges maliciously. Even if the number of corrupted edges is bounded, the adversarial message corruption model can preclude self-stabilizing solutions for most non-trivial problems. It can be easily proved by the standard partition-based argument: consider the MIS problem in a network consisting of two processes (A and B) and one link between the processes. Briefly speaking, a set S of processes in a graph (or a network) is an independent set if any two processes in S are not adjacent to each other, and if S is not a proper subset of any other independent set, we say S is a maximal independent set. In a legitimate configuration where A is independent (i.e., a member of an MIS S) and B is dominated (i.e., a non-member of

S), when A sends a message to inform B that A is an independent process, such message may be corrupted in the link by the adversary and B cannot get the correct information, hence B decides to change its state to become an independent process, which leads to an illegitimate configuration.

The above observation yields the interest in exploring a reasonably relaxed model of message corruption. Probabilistic error models, where message corruption is modeled as a stochastic event that the adversary cannot control, are widely accepted as reasonable assumptions. It is not only standard in information theory but also popular in distributed computing. Self-stabilizing solutions are, however, still ruled out even in most of the probabilistic error-models because it still admits the partition-based argument. More precisely, it allows an execution starting from any legitimate configuration to some non-legitimate one with a non-zero probability.

To circumvent the impossibility of self-stabilization in probabilistic error models, this paper focus on *loose-stabilization* [35], which is a relaxed variant of conventional self-stabilization. While keeping the same convergence property with self-stabilization, loose-stabilization relaxes the closure property: the system is allowed to deviate from legitimate configurations after being legitimate for a long time.

3.1.2 Contributions of This Chapter

In this chapter, we consider loosely-stabilizing solutions for the MIS problem with probabilistic erroneous communications that cannot be solved by self-stabilization. Specifically, we present three systematic approaches for designing loose-stabilizing algorithms in probabilistic error models: the *redundant-state*, the *step-up*, and the *repetition* approaches. All the approaches assume the atomic-state model (where each process can atomically change its state depending on its state and the neighbors' current states) with the uniformly distributed scheduler and follow certain kinds of error-correction/detection mechanisms.

The basic ideas of error-correction/detection mechanisms are straightforward and adopted from information theory and coding theory. In the *redundant-state* approach and the *step-up* approach, we borrow the idea of the hash function. More specifically, we enlarge the domain of the variables of processes. When a process A reads the state of its neighbor B, the value that A gets may be corrupted due to the unreliable communication and become a random value within the domain. Then, process A takes this value as the input and maps it into an output value with a

Table 3.1: Assumptions and Performances of Loosely-Stabilizing MIS Algorithms.

	Redundant-State	Step-Up	Repetition
Process ID	Unavailable	Unavailable	Available
Maximum Degree	Constant	Arbitrary	Arbitrary
Error Distribution	Uniform	Uniform	Arbitrary
$maxECT$	$O(\log n)$	$O(n \log^3 n)$	$O(n^2 \log n)$
$minEHT$	$\Omega(n^d)$	$\Omega(n^d)$	$e^{\Omega(n)}$
Space Complexity	$O(\log n)$	$O(\log n)$	$O(n\Delta \log n)$

range size of two, and A regards the output as the current state of B. In the *repetition* approach, we use the idea of repetition codes: each process repeatedly gets value from its neighbor, counts the occurrences of each obtained value, and regards the majority as the final result. We will discuss the detailed algorithms in Sections 3.3, 3.4, and 3.5.

Two different models of message corruption are considered: *uniformly distributed error* and *arbitrarily distributed error* models. Both models assume that each message is corrupted with a constant probability ρ $(0 < \rho < 1)$. *Uniformly distributed error* model assumes that the values of the corrupted message are uniformly distributed on the message's domain, while in the *arbitrarily distributed error* model there are no such constraints.

We apply these approaches to design three corresponding loosely-stabilizing algorithms for the MIS problem, and the performances of the algorithms are summarized in Table 3.1, where $maxECT$ and $minEHT$ represent the maximum expected convergence time and the minimum expected holding time, respectively, n denotes the number of the processes, d denotes a sufficiently large constant, and Δ denotes the maximum degree among all processes. Notice that, a loosely-stabilizing algorithm is meaningful only if its $minEHT$ is significantly longer than the $maxECT$. The redundant-state and the repetition algorithms have an exponential difference between $maxECT$ and $minEHT$. Although the step-up algorithm does not, the parameter d in $minEHT$ can be set sufficiently large so that the difference between $maxECT$ and $minEHT$ is large enough. In the following, we briefly describe these approaches.

Redundant-State Approach. The *redundant-state* approach is applicable to anonymous networks of bounded degree (i.e., $\Delta = O(1)$) and the uniformly distributed error model with error probability $\rho = 1 - \epsilon$ for an arbitrarily small constant ϵ. The key idea of this approach is to enlarge the domain of the message by introducing a large amount of redundancy so that even if some message is corrupted, it becomes a meaningless message (i.e., the bit sequence is invalid in the correct behavior of the algorithm) with high probability. In other words, the erroneous value in such a message can be detected and does not cause any incorrect effect to the receiver with high probability. This mechanism allows processes to greatly confine the influences from erroneous communications.

Step-Up Approach. To circumvent the constraint of bounded degree in the *redundant-state* approach, the *step-up* approach makes better use of the redundant values of the messages. In the previous approach, we only use the redundant values to reduce the effect of corrupted messages, and we do not care about the exact values in the corrupted messages, while in this approach processes confine the effect of corrupted messages by utilizing the redundant values as the buffer, so that even if a process takes a bad action due to some corrupted message, with high probability the result of such action can be corrected before the neighbors of the process are influenced.

Repetition Approach. The *repetition* approach assumes the arbitrarily distributed error model with error probability $\rho = 1/2 - \epsilon$ for an arbitrarily small constant ϵ. Different from the previous two approaches, the *repetition* approach does not use redundant values. The key idea of this approach is the error-correction by a very simple repetition code with majority decoding. The assumption of $\rho < 1/2 - \epsilon$ admits an error-correction with high probability for a sufficiently large number of repetitions. It should be emphasized that this approach is not so trivial: to utilize block codes (including most of the linear codes with strong error-correction features), some synchronization mechanism is necessary, but our system assumption does not equip with full (round-based) synchrony. It is not clear how one can develop such a mechanism in the loosely-stabilizing manner under the probabilistic scheduler with unreliable communication. Fortunately, the repetition code works without any synchronization mechanism, except for the period around the time when the encoded value changes. Thus it keeps a long holding time after the states of all processes stabilize. In the convergence period, however, the change of processes' states would cause the failure of error correction with an unexpectedly high probability. To prevent such

failure, we install an additional mechanism of priority-based monotone fixing of processes' states. Compared with the error-free case, it yields a longer convergence time, but it is still polynomial of n.

3.2 Specification of MIS Problem

For a given algorithm $\mathcal{A}$, given its *output function* $op_{\mathcal{A}}$ mapping each process-state to 0 or 1, the specification of the MIS problem, denoted by SP_{MIS}, is defined as the following predicate on configurations:

Definition 3.2.1. *For any configuration C, denote $I_{MIS}(C)$ as the set of processes that is in state s such that $op_{\mathcal{A}}(s) = 1$. Then, we define SP_{MIS} as the following: $SP_{MIS}(C) = True$ holds if and only if $I_{MIS}(C)$ is a maximal independent set in G.*

3.3 Redundant-State Approach

3.3.1 Description of Algorithm $\mathcal{RS}$

The algorithm $\mathcal{RS}$ is based on the redundant-state approach, thus it assumes a constant maximum degree (i.e. $\Delta = \Theta(1)$) and the uniformly distributed error model (i.e. $M_{\mathcal{RS}} = M_u$) with error probability $\rho = 1 - \epsilon$ for an arbitrarily small constant ϵ. Processes are anonymous, that is, identifiers are not available to processes. Each process p_i has one variable $v_i \in \{0, 1, ..., c\}$ where the parameter $c = \Theta(n^{d+1})$ and d is a sufficiently large constant. A process sets value c (resp. 0) when it is a member (resp. non-member) of the MIS the algorithm constructs, and values from 1 to $c - 1$ are treated as redundant values. The output function $op_{\mathcal{RS}}$ of $\mathcal{RS}$ is defined as follows: for each process $p_i \in V$, if $v_i = c$ then $op_{\mathcal{RS}}(v_i) = 1$, otherwise $op_{\mathcal{RS}}(v_i) = 0$.

The algorithm $\mathcal{RS}$ (given in Algorithm 3.1) works as follows. Each time a process p_i is activated, it reads the states of its neighbors. Next, if v_i is maximal around its neighbors and v_i is not c, it is set to c; if there exists a neighbor of p_i such that v_i is smaller than that of the neighbor, then v_i is set to 0; if v_i is c and there exists a neighbor of p_i such that the state of the neighbor is also c, then v_i is set to 0.

Take a network consisting of two connected processes p_1 and p_2 as an example. To make the algorithm easier to understand, we assume that initially $0 \leq v_1 < v_2 < c$ and we only consider the case of p_1. When p_1 is activated for the first time, it reads the state of p_2 (which is v_2). If p_1 gets the correct value of v_2 or the value of v_2 is corrupted due to the unreliable communication but still greater than v_1, it will set v_1 to 0 according to Lines 3 and 4 in Algorithm 1; in the other case, v_1 will be set to c according to Lines 1 and 2. Now we consider the latter case where v_1 is set to c, and p_1 is activated for the second time. If p_1 gets the correct value of v_2 or the value of v_2 is corrupted but still less than v_1 (which is equal to c), p_1 does not make any change; in the other case (p_1 incorrectly gets c from p_2), v_1 will be set to 0 according to Lines 5 and 6.

Algorithm 3.1 Redundant-State Algorithm $\mathcal{RS}$ (Behavior of process p_i)

Variables in p_i:

$v_i \in \{0, 1, ..., c\}$

Notation:

$c = \Theta(n^{d+1})$ and d is a sufficiently large constant.

1: **if** $\forall p_j \in Neigh(p_i) : v_j \leq v_i \wedge v_i \neq c$ **then**
2: $\quad v_i \leftarrow c$
3: **if** $\exists p_j \in Neigh(p_i) : v_j > v_i$ **then**
4: $\quad v_i \leftarrow 0$
5: **if** $\exists p_j \in Neigh(p_i) : v_j = c \wedge v_i = c$ **then**
6: $\quad v_i \leftarrow 0$

3.3.2 Analysis of $\mathcal{RS}$

At first, we divide processes into four types according to the states of their own and their neighbors'.

Definition 3.3.1. *(1) independent process: a process p_i is an independent process if $v_i = c$ and $\forall p_j \in Neigh(p_i) : v_j < c$.*

(2) dominated process: a process p_i is a dominated process if $v_i = 0$ and $\exists p_j \in Neigh(p_i) : p_j$ is an independent process.

(3) pseudo-dominated process: a process p_i is a pseudo-dominated process if $v_i = 0$, $\exists p_j \in Neigh(p_i) : v_j = c$, but p_i is not dominated (i.e., there is no independent neighbor).

(4) illegal process: a process p_i is illegal if p_i is not independent, dominated, or pseudo-dominated. That is, (i) $0 < v_i < c$, (ii) $v_i = 0 \wedge \forall p_j \in Neigh(p_i) : v_j < c$, or (iii) $v_i = c \wedge \exists p_j \in Neigh(p_i) : v_j = c$.

Based on Definition 3.3.1, we define the safe configurations of $\mathcal{RS}$:

Definition 3.3.2. *The set of safe configurations $\mathcal{S}_1$ of $\mathcal{RS}$ is the set of configurations where only independent and dominated processes exist.*

Maximum Expected Convergence Time

The basic idea for analyzing the convergence time of the algorithm $\mathcal{RS}$ is to divide the execution of it into two phases. In the first phase, we eliminate all the values of processes' states from 1 to $c - 1$ by setting them to 0 or c by Lemma 1; in the second phase, we prove that there is no illegal or pseudo-dominated process after taking $O(\log n)$ steps with high probability. Now, we consider the first phase.

Lemma 3.3.1. *By taking $\Theta(\log n)$ steps from an arbitrary initial configuration, all processes have value 0 or c with probability $1 - e^{-\Theta(\log n)}$.*

Proof. By the union bound, the probability that every process makes at least one move in $\Theta(\log n)$ steps is at least

$$1 - n \cdot (1 - \phi)^{\Theta(\log n)} = 1 - n \cdot e^{-\Theta(\log n)} = 1 - e^{-\Theta(\log n)}$$

According to Algorithm 1, each process can only change its value to 0 or c when it makes a move. □

By Definition 3 and Lemma 1, all illegal processes are either in the second or third category of the illegal process at the end of the first phase.

In the following, we consider the second phase where the redundant states start to work. Due to the existence of the redundant states, we prove that the algorithm $\mathcal{RS}$ keeps the independent (or dominated respectively) processes staying independent (or dominated respectively) with high probability by Lemmas 2 and 3.

Lemma 3.3.2. *If process p_i is an independent, dominated or pseudo-dominated process, then in any step, the probability that p_i keeps the same v_i is* $1 - O(1/c)$.

Proof. If p_i is an independent process, then $v_i = c$ and $\forall p_j \in Neigh(p_i) : v_j < c$. Process p_i changes v_i to 0 if it is activated and incorrectly obtains c from at least one neighbor. The probability that p_i is activated and incorrectly obtains c from one neighbor is $\phi \cdot \rho/c$. Therefore, the probability that p_i changes v_i to 0 is at most $\phi \cdot \rho \cdot \Delta/c = O(1/c)$ by the union bound. (Remind that $\Delta = \Theta(1)$)

If p_i is a dominated or pseudo-dominated process, then $v_i = 0$ and $\exists p_j \in Neigh(p_i) : v_j = c$. Process p_i changes v_i to c if it is activated and incorrectly obtains 0 from p_j. Such an event happens with probability $\phi \cdot \rho/c = O(1/c)$. □

Lemma 3.3.3. *If process p_i is an independent (or dominated respectively) process, then in any step, the probability that p_i remains independent (or dominated respectively) is* $1 - O(1/c)$.

Proof. If process p_i is an independent process, all its neighbors are dominated processes. Process p_i remains independent if p_i and all its neighbors do not change their values. By Lemma 3.3.2 and the union bound, such an event happens with probability at least $1 - \Delta \cdot O(1/c) = 1 - O(1/c)$.

If process p_i is a dominated process, then $\exists p_j \in Neigh(p_i) : p_j$ is an independent process. One possibility for p_i remaining dominated is that v_i keeps the same and p_j remains independent. By Lemma 3.3.2 and the above result, such an event happens with probability $(1 - O(1/c))^2 = 1 - O(1/c)$. □

Next, we prove that illegal (or pseudo-dominated, respectively) processes become independent (or dominated, respectively) with at least a constant probability by using the assumption of constant degree by Lemmas 4, 5, and 6.

Lemma 3.3.4. *If process p_i is an illegal process, then in any step, the probability that p_i changes v_i is* $\Theta(1)$.

Proof. If the illegal process p_i is in the second category, then $v_i = 0$ and $\forall p_j \in Neigh(p_i) : v_j = 0$. Process p_i changes v_i to c if it is activated and correctly obtains 0 from all its neighbors. such an event happens with probability $\phi \cdot (1 - \rho)^{\Delta_i} = \Theta(1)$.

If the illegal process p_i is in the third category, then $v_i = c$ and $\exists p_j \in Neigh(p_i) : v_j = c$. Denote x to be the number of such neighbors. Process p_i keeps the same v_i if p_i is not activated, or p_i is activated, incorrectly reads all the neighbors that have values c, and does not obtain c from any of its neighbors that have values 0. such an event happens with probability $1 - \phi + \phi \cdot \rho^x \cdot (1 - \rho/c)^{\Delta_i - x} \in [1 - \phi + \phi \cdot \rho^\Delta, 1 - \phi + \phi \cdot \rho]$, which is in $\Theta(1)$. Therefore, the probability that p_i changes v_i is $1 - \Theta(1) = \Theta(1)$. □

Lemma 3.3.5. *If process p_i is an illegal process, then in any step, the probability that p_i becomes independent is* $\Omega(1)$.

Proof. If the illegal process p_i is in the second category, then $v_i = 0$ and $\forall p_j \in Neigh(p_i) : v_j = 0$. Process p_i becomes independent if v_i is changed to c and all its neighbors do not change their values. By Lemmas 3.3.2 and 3.3.4, such an event happens with probability at least

$$\Theta(1) \cdot (1 - \Theta(1))^\Delta = \Theta(1)$$

If the illegal process p_i is in the third category, then $v_i = c$ and $\exists p_j \in Neigh(p_i) : v_j = c$. Process p_i becomes independent if v_i keeps the same, all its neighbors that have values c change their values to 0, and all its neighbors that have values 0 do not change their values. By Lemma 3.3.2 and 3.3.4, such an event happens with probability at least

$$(1 - \Theta(1)) \cdot (\Theta(1))^\Delta \cdot (\Theta(1))^\Delta = \Theta(1)$$

□

Lemma 3.3.6. *If process p_i is a pseudo-dominated process, then in any step, the probability that p_i becomes dominated is* $\Omega(1)$.

Proof. If process p_i is a pseudo-dominated process, then $\exists p_j \in Neigh(p_i) : p_j$ is an illegal process in the third category. One possibility for p_i becoming dominated is that v_i keeps the same and p_j becomes independent. By Lemmas 3.3.2 and 3.3.5, such an event happens with probability $(1 - O(1/c)) \cdot \Omega(1) = \Omega(1)$. □

If the degree is unbounded, the probability that an illegal process p_i in the second or third category becomes independent is exponentially small in the worst case when the values of all

p_i's neighbors are not 0, which yields an exponentially long convergence time. Therefore, the constraint of bounded degree is needed in this algorithm.

Now we consider the configurations after $O(\log n)$ steps from an arbitrary initial configuration. By Lemma 3.3.1, each illegal process is either in the second category or in the third category. Denote the constant $\gamma > 0$ as the lower bound of the probability that an illegal process becomes independent, or a pseudo-dominated process becomes dominated in any step. We can prove that by taking $O(\log n)$ steps there are no illegal or pseudo-dominated processes in the configuration with high probability.

Lemma 3.3.7. *From an arbitrary initial configuration C_0, there is no illegal or pseudo-dominated process after taking $O(\log n)$ steps with probability $1 - O(1/n)$.*

Proof. By Lemmas 3.3.5 and 3.3.6, the probability that an illegal process does not become independent, or a pseudo-dominated process does not become dominated in $\lceil 2\log_{1/(1-\gamma)} n \rceil = O(\log n)$ steps from C_0 is at most $(1-\gamma)^{\lceil 2\log_{1/(1-\gamma)} n \rceil} = O(1/n^2)$. By Lemma 3.3.3, the probability that an independent or dominated process becomes illegal in $O(\log n)$ steps from C_0 is $1 - (1 - O(1/c))^{O(\log n)} \leq O(\log n/c)$. By the union bound, There is no pseudo-dominated or illegal process in $O(\log n)$ steps from C_0 with probability at least $1 - n \cdot O(1/n^2) - n \cdot O(\log n/c) = 1 - O(1/n)$. (Remind that $c = \Theta(n^{d+1})$) □

Combining with the first phase we have the maximum expected convergence time of $\mathcal{RS}$.

Theorem 3.3.1. $\max_{C \in \mathcal{C}_1} ECT_{\mathcal{RS}}(C, \mathcal{S}_1, M_u) = O(\log n)$ *in terms of steps, where $\mathcal{C}_1$ is the set of all possible configurations of $\mathcal{RS}$.*

Proof. By Lemmas 3.3.1 and 3.3.7. □

Minimum Expected Holding Time

Next we analyze the minimum expected holding time of $\mathcal{RS}$.

Theorem 3.3.2. $\min_{C \in \mathcal{S}_1} EHT_{\mathcal{RS}}(C, SP_{MIS}, M_u) = \Omega(n^d)$ *in terms of steps.*

Proof. The probability that at least one process becomes illegal in any step from a safe configuration is at most $n \cdot O(1/c) = O(n/c)$ by Lemma 3.3.3 and the union bound. Hence, the minimum expected holding time is $\Omega(c/n)$ steps, which is $\Omega(n^d)$ steps by $c = \Theta(n^{d+1})$. □

Space Complexity

The space complexity of $\mathcal{RS}$ is straightforward: each process p_i has one variable $v_i \in \{0, 1, ..., c\}$, hence the space complexity is $O(\log c)$, which is $O(\log n)$ bits by $c = \Theta(n^{d+1})$.

Discussion on the parameter d

In Section 3.1 we define the parameter c to be $\Theta(n^{d+1})$ where d is a sufficiently large constant. We let d be sufficiently large to make the analysis easier, while in this section we make a discussion on it and see how this parameter affects the algorithm $\mathcal{RS}$.

In the analysis of maximum expected convergence time (Section 3.2.1), the parameter d does not influence the results of the lemmas except Lemma 7, where d must be greater than or equal to 2 to keep the same result with high probability. In Section 3.2.2, we know that the minimum expected holding time is $\Omega(n^d)$ steps, hence the choice of d directly affects the result: the larger d is, the longer minimum expected holding time we have. However, d also has direct influence on the memory space in processes. Therefore, we can adjust d to make trade-offs between the minimum expected holding time and the memory space, while $d \geq 2$.

3.4 Step-Up Approach

3.4.1 Description of Algorithm $\mathcal{SU}$

The algorithm $\mathcal{SU}$ is based on the step-up approach that has the same setting as the redundant-state approach, and circumvents the constraint of bounded degree, thus we consider the graphs that $\mathcal{SU}$ works on have no restriction on the degree of processes. The output function $op_{\mathcal{SU}}$ of $\mathcal{SU}$ is defined as follows: for each process $p_i \in V$, if $v_i = c$ then $op_{\mathcal{SU}}(v_i) = 1$, otherwise $op_{\mathcal{SU}}(v_i) = 0$.

The algorithm $\mathcal{SU}$ (given in Algorithm 2) works as follows. Each time a process p_i is activated, it reads the states of its neighbors. Next, if p_i has at least one neighbor whose state is c, then v_i is set to 0; in the other case v_i takes a step up: it is increased by $\lfloor c/\log^2 c \rfloor$ unless v_i exceeds the domain, and if that happens v_i is set to c. Notice that in $\mathcal{SU}$ we do not treat the states from 1 to $c - 1$ as the contaminated values, but as the buffer that the processes can utilize to confine

the effect of erroneous communications in the following way: even if a process misrecognizes that no neighbor of it is a member of the MIS due to some erroneous communication, the process gradually approaches to the state of an MIS member instead of instantly becoming an MIS member. This enables the process to correct its state with high probability before becoming an MIS member.

Take a network consisting of two connected processes p_1 and p_2 as an example. To make the algorithm easier to understand, we assume that initially $0 \leq v_1 < v_2 < c$ and we only consider the case of p_1. When p_1 is activated for the first time, it reads the state of p_2 (which is v_2). If p_1 gets the correct value of v_2 or the value of v_2 is corrupted due to the unreliable communication but still less than c, it will increase v_1 by $\lfloor c/\log^2 c \rfloor$ according to Lines 9 and 10 in Algorithm 2; in the other case (p_1 incorrectly gets c from p_2), v_1 will be set to 0 according to Lines 7 and 8. If p_1 continuously reads p_2 correctly for $O(\log^2 c)$ times, v_1 will be increased to c.

Algorithm 3.2 Step-Up Algorithm $\mathcal{SU}$ (Behavior of process p_i)

Variables in p_i: $v_i \in \{0, 1, ..., c\}$

Notation: $c = \Theta(n^{d+1})$ and d is a sufficiently large constant.

1: **if** $\exists p_j \in Neigh(p_i) : v_j = c$ **then**
2: $\quad v_i \leftarrow 0$
3: **if** $\forall p_j \in Neigh(p_i) : v_j \neq c$ **then**
4: $\quad v_i \leftarrow \min\{c, v_i + \lfloor c/\log^2 c \rfloor\}$

3.4.2 Analysis of $\mathcal{SU}$

Processes in $\mathcal{SU}$ are divided into three types according to the states of their own and their neighbors'.

Definition 3.4.1. *(1) independent process: a process p_i is an independent process if $v_i = c$ and $\forall p_j \in Neigh(p_i) : v_j \neq c$. Further, process p_i is a safely independent process if $v_i = c$ and $\forall p_j \in Neigh(p_i) : |v_j - c| = \omega(c/\log c)$.*
(2) dominated process: a process p_i is a dominated process if $v_i \neq c$ and $\exists p_j \in Neigh(p_i) : v_j = c$.

(3) illegal process: a process is an illegal process if it does not belong to any of the above two types of processes.

Based on Definition 5, we define the potential set $Pot(C)$ in each configuration C.

Definition 3.4.2. *The potential set $Pot(C)$ concerning the configuration C is the set of safely independent processes and all their neighbors in the configuration C.*

Based on the definition of the potential set, we define the set of safe configurations $\mathcal{S}_2$ of $\mathcal{SU}$.

Definition 3.4.3. *The set of safe configurations $\mathcal{S}_2$ of $\mathcal{SU}$ is the set of configurations C where $Pot(C) = V$.*

Minimum Expected Holding Time

In a safe configuration, there only exist safely independent processes and dominated processes whose values are low enough by Definition 5, thus these are the processes that we need to concern about now. Similar to the algorithm $\mathcal{RS}$, we prove that each safely independent process in $\mathcal{SU}$ does not change its values with high probability due to the existence of the redundant states by Lemma 8.

Lemma 3.4.1. *In any step, an independent process p_i keeps the same v_i with probability* $1 - O(\Delta/c)$.

Proof. Since p_i is an independent process, we have $v_i = c$ and $\forall p_j \in Neigh(p_i) : v_j \neq c$. Process p_i keeps the same v_i if p_i is not activated, or p_i is activated and does not obtain c from any of its neighbors. such an event happens with probability

$$1 - \phi + \phi \cdot (1 - \rho/c)^{\Delta_i} = 1 - O(\Delta/c)$$

□

As for each dominated process, it increases its value when it incorrectly thinks that the values of all the neighbors are not c, and we prove that such an incorrect move happens with a constant probability by Lemma 9.

Lemma 3.4.2. *In any step, a dominated process p_i increases v_i with probability at most $\phi \cdot \rho$, or sets v_i to 0 with probability at least $\phi(1-\rho)$.*

Proof. Since p_i is a dominated process, we have $v_i \neq c$ and $\exists p_j \in Neigh(p_i) : v_j = c$. Process p_i increases v_i if it is activated and incorrectly reads p_j. Such an event happens with probability $\phi \cdot \rho$. Process p_i keeps the same v_i if p_i is not activated, hence the probability is $1-\phi$. Consequently, the probability that p_i sets v_i to 0 is at least $1-(1-\phi)-\phi \cdot \rho = \phi(1-\rho)$. □

For a dominated process in the safe configurations, its value is low enough so that $O(\log c)$ incorrect moves are needed to make it illegal, as well as its safely independent neighbors. Such a gap (the difference between the values of a safely independent process and a dominated process) works as a buffer that makes dominated processes illegal only with an extremely small probability of $\Theta(1)^{\Omega(\log c)} = e^{-\Omega(\log c)}$ by Lemma 9. Combining with Lemma 8, we can conclude that the minimum expected holding time of $\mathcal{SU}$ is $\min\{\Omega(c/\Delta), e^{\Omega(\log c)}\} = \Omega(c/\Delta)$ steps, which is $\Omega(n^{d+1}/\Delta) = \Omega(n^d)$ steps by the choices of the parameter c and d.

Theorem 3.4.1. $\min_{C \in \mathcal{S}_2} EHT_{\mathcal{SU}}(C, SP_{MIS}, M_u) = \Omega(n^d)$ *in terms of steps.*

Maximum Expected Convergence Time

To analyze the expected convergence time of $\mathcal{SU}$, we first see how illegal processes act in the executions by Lemmas 10 and 11.

Lemma 3.4.3. *In any step, an illegal process p_i where $v_i = c$ keeps the same v_i with a constant probability.*

Proof. Since process p_i is an illegal process where $v_i = c$, we have $\exists p_j \in Neigh(p_i) : v_j = c$, denote δ be the number of such neighbors. Process p_i keeps the same v_i if it is not activated, or it is activated and does not obtain c from any of its neighbors. such an event happens with probability $1-\phi+\phi \cdot \rho^{\delta}(1-\rho/c)^{\Delta_i-\delta}$, which is at least $1-\phi$ and at most $1-\phi+\phi \cdot \rho$. □

Lemma 3.4.4. *In any step, an illegal process p_i where $v_i < c$ increases v_i with probability at least $\phi/2$, or sets v_i to 0 with probability $O(\Delta/c)$.*

Proof. Since p_i is an illegal process where $v_i \neq c$, we have $\forall p_j \in Neigh(p_i) : v_j \neq c$. Process p_i increases v_i if it is activated and does not obtain c from any of its neighbors. such an event happens with probability $\phi \cdot (1 - \rho/c)^{\Delta_i} \geq \phi/2$.

Process p_i sets v_i to 0 if it is activated and incorrectly obtains c from at least one of its neighbors. By the union bound, such an event happens with probability at most $\phi \cdot \Delta \cdot \rho/c = O(\Delta/c)$.

□

Next, we define a set of processes that have *high* values as $High(C)$ in each configuration C. Processes in this set can be seen as good candidates to become safely independent processes in subsequent configurations.

Definition 3.4.4. *Define $High(C) \subseteq V - Pot(C)$ to be the set of processes p_i which have values v_i in configuration C such that $|v_i - c| = O(c/\log c)$.*

The execution of Algorithm $\mathcal{SU}$ consists of multiple iterations. In each iteration, we guarantee that the number of safely independent processes is incremented by at least one with high probability, hence after $O(n)$ iterations a safe configuration S is reached, i.e., $Pot(S) = V$, with high probability. The end of each iteration is the configuration where the number of safely independent processes is incremented by at least one from the previous configuration, and it is also the beginning of the next iteration. The beginning of the first iteration is the initial configuration.

First, we prove that for any process p_i that is not in $Pot(C_0)$ where C_0 is an arbitrary configuration, by taking $O(\log^2 c)$ steps it becomes either a safely independent process or a good candidate of becoming a safely independent process with high probability by Lemma 12.

Lemma 3.4.5. *From an arbitrary configuration C_0, a configuration C_1 where $\exists p_i \in V - Pot(C_0) : p_i \in Pot(C_1) \cup High(C_1)$ is reached in $O(\log^2 c)$ steps with high probability.*

Proof. At first, the lemma holds trivially in the case where $\exists p_i \in V - Pot(C_0) : v_i = c$ in C_0.

Now consider the case where $\forall p_i \in V - Pot(C_0) : v_i \neq c$ in C_0. Let $p_j \in V - Pot(C_0)$ be a process where $v_j = c$ in C_1. Process p_j increases v_j to c from C_0 to C_1 if it increases v_j for $O(\log^2 c)$ times, and does not set v_j to 0 for once during C_0 to C_1. Denote X be the number of

times that p_j increases v_j in $(4\log^2 c)/\phi$ steps, then X satisfies binomial distribution. By the Chernoff Bound and Lemma 3.4.4, we have

$$Pr(X \leq \log^2 c) = Pr(X \leq \mathbf{E}[X]/2) \leq e^{-\mathbf{E}[X]/8} = e^{-O(\log^2 c)}$$

Thus, the probability that $X \geq \log^2 c$ is at least $1 - e^{-O(\log^2 c)}$. The probability that p_j does not set v_j to 0 for once in $O(\log^2 c)$ steps is at least $1 - O((\log^2 c) \cdot \Delta/c)$ by Lemma 3.4.4. Therefore, we have $v_j = c$ in C_1 with high probability. By Definitions 3.4.2 and 3.4.4, if $v_j = c$ in C_1, then $p_j \in Pot(C) \cup High(C)$, which leads to the lemma. □

By the analysis of the minimum expected holding time in Section 4.2.1, we know that processes in the potential set do not change their states with high probability, hence we only consider the processes in $High(C)$ next. In the following lemma, we prove that for any process p_i in $High(C_0)$ where C_0 is an arbitrary configuration, by taking $O(\log c)$ steps it is in the potential set with at least constant probability.

Lemma 3.4.6. *From a configuration C_0 where $High(C_0) \neq \emptyset$, a configuration C_1 where $\exists p_i \in High(C_0)$ such that $p_i \in Pot(C_1)$ is reached in $O(\log c)$ steps with at least a constant probability.*

Proof. First, we prove the probability Pr_{C_0} that $\forall p_i \in High(C_0) : p_i$ sets v_i to 0 in $O(\log c)$ steps is at most a constant probability. Denote p_L as the last process in $High(C_0)$ that sets its value to 0, and C_L as the configuration when p_L does so. There are two cases for p_L in C_L:

(1): $p_L \in High(C_L)$ and none of p_L's neighbors is in $High(C_L)$. In this case, p_L is an independent process or an illegal process with value less than c. Thus, the probability that p_L sets v_L to 0 in C_L is $O(\Delta/c)$ by Lemmas 3.4.1 and 3.4.4. Therefore,

$$\begin{aligned} Pr_{C_0} &= Pr_{others} \cdot Pr_L \\ &\leq 1 \cdot O(\Delta/c) \\ &\leq \Theta(1) \end{aligned}$$

where Pr_{others} denotes the probability that other processes in $High(C_0)$ except p_L set their values to 0 before p_L in $O(\log c)$ steps, and Pr_L denotes the probability that p_L sets v_L to 0 in $O(\log c)$ steps.

(2): $p_L \in High(C_L)$ and there exists at least one neighbor of p_L that is also in $High(C_L)$. Denote $p_{L_1}, p_{L_2}, ...$ as these neighbors. Therefore, $p_L, p_{L_1}, p_{L_2}, ...$ set their values to 0 at the same time (in configuration C_L). Without loss of generality, let p_{L_1} be the process that has the largest value among $p_{L_1}, p_{L_2}, ...$ Consider processes p_L and p_{L_1}. There are four possible combinations of values for these two processes: $v_L = v_{L_1} = c$, $v_L = c$ and $p_{L_1} \in High(C_L)$, $p_L \in High(C_L)$ and $v_{L_1} = c$, and $p_L, p_{L_1} \in High(C_L)$. In either combination, by Lemmas 3.4.1, 3.4.2, 3.4.3 and 3.4.4 we have

$$\begin{aligned} Pr_{C_0} &= Pr_{others} \cdot Pr_L \\ &\leq 1 \cdot \Theta(1) \\ &\leq \Theta(1) \end{aligned}$$

Similarly, denote Pr_{others} as the probability that other processes in $High(C_0)$ except p_L and p_{L_1} set their values to 0 before or at the same time with p_L in $O(\log c)$ steps, and Pr_L denotes the probability that p_L sets v_L to 0 in $O(\log c)$ steps.

Therefore, the probability that $\exists p_i \in High(C_0) : p_i$ does not set v_i to 0 in $O(\log c)$ steps is at least $\Theta(1)$. By using the similar argument in Lemma 3.4.5, for each such p_i, we have $v_i = c$ in C_1 with high probability. By Lemmas 3.4.1, 3.4.2, 3.4.3 and 3.4.4, all neighbors of p_i set their values to 0 for at least once during C_0 to C_1 with probability at least $\left(1 - \Theta(1)^{O(\log c)}\right)^{\Delta_i} \geq 1 - e^{-\Theta(\log c)}$. Therefore, the probability that p_i is a safely independent process in C_2 is at least constant. □

Next, we prove that by taking $O(\log n \cdot \log^2 c)$ steps from the beginning of an iteration, the number of safely independent processes is incremented by at least one with high probability, and after $O(n)$ iterations a safe configuration S is reached, i.e., $Pot(S) = V$, with high probability by Lemmas 14 and 15 respectively.

Lemma 3.4.7. *From an arbitrary initial configuration C_0, a configuration C_1 where $|Pot(C_1)| - |Pot(C_0)| \geq 1$ is reached in $O(\log n \cdot \log^2 c)$ steps with high probability.*

Proof. By Lemma 3.4.5 and 3.4.6, we have that $\exists p_i \in V - Pot(C_0) : p_i$ is a safely independent process with at least a constant probability in $O(\log^2 c)$ steps from C_0. Therefore, the probability that $\exists p_i \in V - Pot(C_0) : p_i$ is a safely independent process in C_1 is at least $1 - e^{-\Theta(\log n)}$. □

Lemma 3.4.8. *From an arbitrary configuration C_0, a configuration C_1 where $Pot(C_1) = V$ is reached in $O(n \log n \cdot \log^2 c)$ steps with high probability.*

Proof. By Lemma 3.4.1, the probability that each safely independent process p_i keeps the same v_i in $O(n \log n \cdot \log^2 c)$ steps is

$$\left(1 - O\left(\frac{\Delta}{c}\right)\right)^{O(n \log n \cdot \log^2 c)} \geq 1 - O\left(\frac{\Delta n \log n \cdot \log^2 c}{c}\right)$$

By Lemma 3.4.2, for each dominated process p_j that is neighboring to a safely independent process, the probability that p_j keeps increasing v_j for $\Theta(\log c)$ steps is at most $(\phi \cdot \rho)^{\Theta(\log c)} = e^{-\Theta(\log c)}$, hence the probability that v_j is set to 0 for at least once in $\Theta(\log c)$ steps is at least $1 - e^{-\Theta(\log c)}$. Therefore, the probability that $|v_j - c| = \omega(c/\log c)$ in $O(n \log n \cdot \log^2 c)$ steps is at least

$$\left(1 - e^{-\Theta(\log c)}\right)^{O(n \log n \cdot \log c)} \geq 1 - e^{-\Theta(\log c)}$$

Combining with Lemma 3.4.7 we have this lemma. □

By the choice of the parameter c, we have the following theorem.

Theorem 3.4.2. $\max_{C \in \mathcal{C}_2} ECT_{\mathcal{SU}}(C, \mathcal{S}_2, M_u) = O(n \log^3 n)$ *in terms of steps, where $\mathcal{C}_2$ is the set of all possible configurations of $\mathcal{SU}$.*

Compared with the previous algorithm $\mathcal{RS}$, illegal processes in $\mathcal{SU}$ need to take a non-negligible number of "steps" to become independent. Consequently, the convergence time is longer than that of $\mathcal{RS}$. However, by using this mechanism, we can remove the constraint of bounded degree in $\mathcal{RS}$.

Space Complexity

Processes in $\mathcal{SU}$ have the same variables as those in $\mathcal{RS}$, thus the space complexity of $\mathcal{SU}$ is also $O(\log n)$ bits.

3.5 Repetition Approach

3.5.1 Description of Algorithm $\mathcal{RE}$

The algorithm $\mathcal{RE}$ is based on the repetition approach, it requires a unique identifier ID_i for each process p_i to break the symmetry, and each process p_i has a binary variable $v_i \in \{0, 1\}$. The algorithm $\mathcal{RE}$ assumes the arbitrarily distributed error model (i.e. $M_{\mathcal{RE}} = M_a$) with error probability $\rho < 1/2 - \epsilon$ for an arbitrarily small constant ϵ. This assumption is made because by repeatedly reading states from neighboring processes and selecting the majority value as the final result, the algorithm enables error-correction with a high probability. Further analysis of this probability will be provided in subsequent sections. It is worth noting that an alternative assumption can be made with $\rho > 1/2 + \epsilon$, in which case processes would select the minority. However, since all processes share the same algorithm, a choice must be made between the two options. The output function $op_{\mathcal{RE}}$ of $\mathcal{RE}$ is defined as follows: for each process $p_i \in V$, if $v_i = 1$ then $op_{\mathcal{RE}}(v_i) = 1$, otherwise $op_{\mathcal{RE}}(v_i) = 0$.

The fundamental idea of this algorithm is first to select the processes with the largest IDs among their neighbors as the independent processes, then they inform their states to neighbors and the neighbors become dominated processes. Next, also select the processes with the largest IDs among their neighbors except for the dominated processes as the independent processes, then they do the same thing as the processes in the first step, and so on. Due to the unreliable communication, processes inform their states to neighbors for multiple times, and the neighboring processes regard the majority as the final result to increase the probability of getting the true information.

The algorithm $\mathcal{RE}$ (given in Algorithm 3) works as follows. For each process p_i, if the counter $counter_i$ that represents the number of readings p_i makes so far is less than $2k + 1$ where the parameter $k = \Theta(n)$, it obtains ID_j and v_j from each of its neighbor p_j, stores the ID_j in a dictionary $id\_cand_i(j)$ with the entry $id\_cand_i(j)[ID_j]$, increments another counter $counter\_one_i(j)$ if $v_j = 1$, which represents the number of value 1 p_i obtains from p_j, and increments $counter_i$. If $counter_i = 2k + 1$, process p_i updates v_i in the following way: at first, for each $p_j \in Neigh(p_i)$, process p_i regards the key with the largest value in the dictionary $id\_cand_i(j)$ as true ID_j, and updates the dictionary id_table_i whose keys are local identifiers for all the neighbors of p_i; besides, process p_i regards $v_j = 1$ if $counter\_one_i(j) > k$. Next, if

$\exists p_j \in Neigh(p_i)$ such that $ID_j \geq ID_i$ and $v_j = 1$, process p_i sets v_i to 0; in other cases p_i sets v_i to 1. At last, process p_i resets all the counters and dictionaries id_cand_i.

Take a network consisting of two connected processes p_1 and p_2 as an example. Without loss of generality, we assume that $ID_1 > ID_2$. To make the algorithm easier to understand, we only consider the case of p_1 and we assume that initially $v_1 = v_2 = 0$, $counter_1 = 0$, $counter\_one_1(2) = 0$, $id\_cand_1(2) = \{\}$ (an empty dictionary), and $id\_table_1 = \{p_2 : ID_{init}\}$ (ID_{init} is arbitrary). When p_1 is activated for the first time, we execute Lines 11 to 20 in Algorithm 3: p_1 increments $counter_1$ by one, reads ID_2 and v_2 from p_2, puts ID_2 into the dictionary $id\_cand_1(2)$ where the key is ID_2 and the value is the occurrence of corresponding key, and increments $counter\_one_1(2)$ by one if p_1 incorrectly regards the value of v_2 as 1. If p_1 correctly reads the state of p_2 at this step, the state of p_1 will be updated to $v_1 = 0$, $counter_1 = 1$, $counter\_one_1(2) = 0$, $id\_cand_1(2) = \{ID_2 : 1\}$, and $id\_table_1 = \{\}$.

When p_1 is activated for $2k+1$ times (which means that $counter_1 = 2k+1$), we execute Lines 21 to 31: p_1 regards the key (denote as ID'_2) with the maximum value in $id\_cand_1(2)$ as the ID of p_2 and updates id_table_1 into $id\_table_1 = \{p_j : ID'_2\}$. If $ID'_2 = ID_2$ or ID'_2 is erroneous but still less than ID_1, p_1 sets v_1 to 1 according to Lines 26 and 27; if $ID'_2 \geq ID_1$ (which also means ID'_2 is erroneous), and $counter\_one_1(2) > k$ (which means p_1 reads v_2 as 1 for more than k times), p_1 sets v_1 to 0 according to Lines 24 and 25. After that, p_1 initializes its state by setting $counter_1$ to 0, $counter\_one_1(2)$ to 0, and $id\_cand_1(2)$ to empty.

3.5.2 Analysis of $\mathcal{RE}$

Since each process p_i regards the majority value among the results of $2k + 1$ reads from each neighbor p_j, we expect that such majority is the true value of p_j with high probability. However, it does not hold if p_j changes its state during the communication period. Therefore, we try to *fix* the state of a process gradually by introducing the notion of a *maximal process*.

Definition 3.5.1. *Process p_i is a maximal process in G if $\forall p_j \in Neigh(p_i) : ID_j < ID_i$ holds in G.*

Based on Definition 3.5.1, we inductively divide processes into some levels.

Algorithm 3.3 Repetition Algorithm $\mathcal{RE}$ (Behavior of process p_i)

Constants in p_i: ID_i

Variables in p_i:

$v_i \in \{0, 1\}$

$counter_i \in \{0, 1, 2, ..., 2k + 1\}$

$counter\_one_i(j) \in \{0, 1, 2, ..., 2k + 1\}$

$id\_cand_i(j)$: a dictionary where keys are ID_js read from $p_j \in Neigh(p_i)$

id_table_i: a list indexed on the neighbors of p_i

1: **if** $counter_i < 2k + 1$ **then**
2: $counter_i \leftarrow counter_i + 1$
3: **for** each $p_j \in Neigh(p_i)$ **do**
4: **if** $ID_j \in id\_cand_i(j)$ **then**
5: $id\_cand_i(j)[ID_j] \leftarrow id\_cand_i(j)[ID_j] + 1$
6: **else**
7: create key ID_j in $id\_cand_i(j)$
8: $id\_cand_i(j)[ID_j] \leftarrow 1$
9: **if** $v_j = 1$ **then**
10: $counter\_one_i(j) \leftarrow counter\_one_i(j) + 1$
11: **end for**
12: **if** $counter_i = 2k + 1$ **then**
13: **for** each $p_j \in Neigh(p_i)$ **do**
14: $id\_table_i[j] \leftarrow \arg\max_{key} id\_cand_i(j)[key]$
15: **end for**
16: **if** $\exists p_j \in Neigh(p_i) : id\_table_i[j] \geq ID_i \wedge counter\_one_i(j) > k$ **then**
17: $v_i \leftarrow 0$
18: **else**
19: $v_i \leftarrow 1$
20: $counter_i \leftarrow 0$
21: **for** each $p_j \in Neigh(p_i)$ **do**
22: $counter\_one_i(j) \leftarrow 0$
23: $id\_cand_i(j) \leftarrow$ empty
24: **end for**

Definition 3.5.2. *Define $M(1)$ to be the set of maximal processes in G, and $N(1)$ to be the set of processes in G that are neighboring to any process in $M(1)$. We say that processes in $M(1)$ and $N(1)$ are in the first level. For level $x \geq 2$, define $M(x)$ to be the set of maximal processes in the induced sub-graph $G_x = G\left[V - \bigcup_{y=1}^{x-1}(M(y) \cup N(y))\right]$, and $N(x)$ to be the set of processes in G_x that are neighboring to any process in $M(x)$.*

To define the safe configurations of $\mathcal{RE}$, we first define the set of independent and dominated processes in Definition 11, and then introduce the notion of *legitimate initial state* for each process in Definition 12.

Definition 3.5.3. *(1) independent process: a process p_i is an independent process if $v_i = 1$ and $\forall p_j \in Neigh(p_i) : v_j = 0$. (2) dominated process: a process p_i is a dominated process if $v_i = 0$ and $\exists p_j \in Neigh(p_i) : v_j = 1$.*

Definition 3.5.4. *Process p_i is in the legitimate initial state if $counter_i = 0$ and $\forall p_j \in Neigh(p_i) : counter\_one_i(j) = 0$ and $id\_cand_i(j) = \emptyset$.*

In the following two lemmas we prove that each process can reach its legitimate initial state in $O(k \log n)$ steps with high probability.

Lemma 3.5.1. *In every $\Theta(\log n)$ steps, each process is activated at least once with probability $1 - e^{-\Theta(\log n)}$.*

Proof. By the union bound, the probability that every process makes at least one move in $\Theta(\log n)$ steps is at least

$$1 - n \cdot (1 - \phi)^{\Theta(\log n)} = 1 - n \cdot e^{-\Theta(\log n)} = 1 - e^{-\Theta(\log n)}$$

□

Lemma 3.5.2. *From an arbitrary initial configuration, process p_i reaches its legitimate initial state in $\Theta(k \log n)$ steps with probability $1 - e^{-\Theta(\log n)}$.*

Proof. According to Algorithm 3, process p_i resets all its counters and id_cand_i as long as the counter $counter_i$ reaches $2k+1$. It takes at most $2k+2$ moves for p_i to do so from an arbitrary initial

state. By Lemma 3.5.1, process p_i makes at least $2k+2$ moves in $(2k+2)\cdot\Theta(\log n) = \Theta(k\log n)$ steps with probability at least

$$(1-e^{-\Theta(\log n)})^{2k+2} \geq 1-(2k+2)\cdot e^{-\Theta(\log n)} = 1-e^{-\Theta(\log n)}$$

□

We focus on the execution after each process experiences a legitimate initial state and define the safe configurations of $\mathcal{RE}$ based on Definitions 3.5.3 and 3.5.4.

Definition 3.5.5. *The set of safe configurations $\mathcal{S}_3$ of $\mathcal{RE}$ is the set of configurations where $\forall p \in \bigcup_{y=1}^{n} M(y)$: p is an independent process, $\forall q \in \bigcup_{y=1}^{n} N(y)$: q is a dominated process.*

Maximum Expected Convergence Time

The basic idea for analyzing the convergence time of $\mathcal{RE}$ is to inductively analyze the states of processes in each level. First, we prove that with high probability, each process can eventually get the true IDs of its neighbors by Lemma 18.

Lemma 3.5.3. *For any process p_i, from its legitimate initial state, the list id_table_i stores the true IDs of all its neighbors in $\Theta(k\log n)$ steps with probability $1-e^{-\Theta(\log n)}$.*

Proof. When p_i is in the legitimate initial state, all its counters are 0 and all its id_cand_i are empty. It takes $2k+2$ moves for p_i to obtain a new id_table_i. By the same argument in Lemma 3.5.2, we know that process p_i makes at least $2k+2$ moves in $\Theta(k\log n)$ steps with probability at least $1-e^{-\Theta(\log n)}$.

Consider the situation where process p_i repeatedly reads a variable var of $p_j \in Neigh(p_i)$. At each time of reading, the result is incorrect with probability $\rho < 1/2-\epsilon$ where ϵ is an arbitrarily small constant. After reading for $2k+1$ times, process p_i regards the majority of the results as the true value of var. Therefore, if p_i incorrectly reads var for at most k times, it can obtain the true value of var. such an event happens with probability

$$\sum_{i=0}^{k}\binom{2k+1}{i}\rho^i(1-\rho)^{2k+1-i}$$

Now we analyze the lower bound and the upper bound of the above probability. By applying Hoeffding's inequality, we have

$$\begin{aligned}
&\sum_{i=0}^{k} \binom{2k+1}{i} \rho^i (1-\rho)^{2k+1-i} \\
=&1 - \sum_{i=0}^{k} \binom{2k+1}{i} (1-\rho)^i \rho^{2k+1-i} \\
\geq&1 - exp(-2(2k+1)(1-\rho-\frac{k}{2k+1})^2)
\end{aligned}$$

Since $\rho \in [0, \frac{1}{2} - \epsilon]$ for any small constant ϵ, $1 - \rho - \frac{k}{2k+1}$ is $\Omega(1)$. Therefore,

$$\sum_{i=0}^{k} \binom{2k+1}{i} \rho^i (1-\rho)^{2k+1-i} \geq 1 - e^{-\Theta(k)}$$

By applying approximated Stirling's formula, we have

$$\begin{aligned}
&\sum_{i=0}^{k} \binom{2k+1}{i} \rho^i (1-\rho)^{2k+1-i} \\
=&1 - \sum_{i=0}^{k} \binom{2k+1}{i} (1-\rho)^i \rho^{2k+1-i} \\
=&1 - \frac{1}{\sqrt{2(2k+1)}} \cdot \\
&\exp\left(-k \log \frac{k(k+1)}{\rho(1-\rho)(2k+1)^2} - \log \frac{k+1}{\rho(2k+1)}\right)
\end{aligned}$$

Since $\rho \in [0, \frac{1}{2} - \epsilon]$ for any small constant ϵ, and $\frac{k}{2k+1}$ is asymptotically equal to $\frac{1}{2}$, $\log \frac{k(k+1)}{\rho(1-\rho)(2k+1)^2}$ and $\log \frac{k+1}{\rho(2k+1)}$ is $O(1)$ and greater than 0. Therefore,

$$\begin{aligned}
\sum_{i=0}^{k} \binom{2k+1}{i} \rho^i (1-\rho)^{2k+1-i} &\leq 1 - \frac{1}{\sqrt{2(2k+1)}} e^{-\Theta(k)} \\
&= 1 - e^{-\Theta(k)}
\end{aligned}$$

Since the lower bound and the upper bound are equal, we have

$$\sum_{i=0}^{k} \binom{2k+1}{i} \rho^i (1-\rho)^{2k+1-i} = 1 - e^{-\Theta(k)}$$

Therefore, the majority of $2k+1$ results that p_i obtains from $p_j \in Neigh(p_i)$ is correct with probability $1-e^{-\Theta(k)}$, and the probability that p_i obtains the correct id_table_i is $(1-e^{-\Theta(k)})^{\Delta_i} = 1 - e^{-\Theta(k)}$. (Remind that $k = \Theta(n)$). Together we have the lemma. □

Obviously, for any integer $x \geq 2$, if $V - \bigcup_{y=1}^{x-1} (M(y) \cup N(y))$ is not empty, then $M(x) \cup N(x) \neq \emptyset$. Therefore, we have $\bigcup_{y=1}^{n} (M(y) \cup N(y)) = V$. As a consequence, all processes are gradually fixed into the corresponding sets based on the IDs. At the same time, the state of each process is also gradually fixed with high probability by Algorithm $\mathcal{RE}$, starting from the processes in $M(1)$ and expanding into $N(1), M(2), N(2), \cdots$. In this manner, a *fixed* process p_i can change its state only if the majority result of p_i's reading from one of its *fixed* neighbors, say p_j, becomes a wrong value, and we prove that such an event happens with a negligible probability by Lemma 19.

Lemma 3.5.4. *After taking $O(k \log n)$ steps from an arbitrary initial configuration, with probability $1 - e^{-\Theta(\log n)}$, all processes in $M(1)$ are independent processes, and all processes in $N(1)$ are dominated processes. Moreover, all processes in $M(1)$ and $N(1)$ does not change their type in each subsequent $\Theta(k \log n)$ steps with probability $1 - e^{-\Theta(\log n)}$.*

Proof. By Lemmas 3.5.2, 3.5.3 and the union bound, each process $p_i \in M(1)$ is aware of that it is a maximal process and sets v_i to 1 in $\Theta(k \log n)$ steps from an arbitrary initial configuration with probability at least $1 - n \cdot e^{-\Theta(\log n)} = 1 - e^{-\Theta(\log n)}$. After that, each process $p_j \in Neigh(p_i)$ is aware of that $ID_j < ID_i$ and $v_i = 1$, and sets v_j to 0 in another $\Theta(k \log n)$ steps with probability $1 - e^{-\Theta(\log n)}$.

In each subsequent $\Theta(k \log n)$ steps, each process p_i in $M(1)$ or $N(1)$ keeps the same id_table_i and v_i with probability $1 - e^{-\Theta(\log n)}$, thus p_i does not change its type. □

By using almost the same argument we can obtain the following lemma for any $x \geq 2$.

Lemma 3.5.5. *If $M(x)$ is not empty, after taking $O(kx \log n)$ steps from an arbitrary initial configuration, with probability $1 - e^{-\Theta(\log n)}$, all processes in $M(x)$ are independent processes,*

and all processes in $N(x)$ are dominated processes. Moreover, all processes in $M(x)$ and $N(x)$ does not change their type in each subsequent $O(k \log n)$ steps with probability $1 - e^{-\Theta(\log n)}$.

Theorem 3.5.1. $\max_{C \in \mathcal{C}_3} ECT_{\mathcal{RE}}(C, \mathcal{S}_3, M_a) = O(n^2 \log n)$ *in terms of steps, where $\mathcal{C}_3$ is the set of all possible configurations of $\mathcal{RE}$.*

Proof. By Lemma 3.5.5, in $O(kn \log n)$ steps from an arbitrary initial configuration, all processes in $\bigcup_{y=1}^{n} M(y)$ are independent processes, and all processes in $\bigcup_{y=1}^{n} N(y)$ are dominated processes with probability $1 - e^{-O(\log n)}$. Since $\bigcup_{y=1}^{n} (M(y) \cup N(y)) = V$, we have $\bigcup_{y=1}^{n} M(y)$ is an MIS. Together with $k = \Theta(n)$, we have the theorem. □

Minimum Expected Holding Time

From a safe configuration, the necessary condition for specification SP_{MIS} to be violated is that a process p_i obtains an incorrect value of a variable from $p_j \in Neigh(p_i)$, and the probability that such a case happens is $e^{-O(k)}$. Together with $k = \Theta(n)$, the maximum expected holding time is $\Theta(k)/e^{-O(k)} = e^{\Omega(n)}$

Theorem 3.5.2. $\min_{C \in \mathcal{S}_3} EHT_{\mathcal{RE}}(C, SP_{MIS}, M_a) = e^{\Omega(n)}$ *in terms of steps.*

Space Complexity

For each process p_i, the most space consuming variable is the dictionary id_cand_i. For each $p_j \in Neigh(p_i)$, there is a corresponding $id\_cand_i(j)$; each $id\_cand_i(j)$ stores at most n keys, and the value for each key is at most $2k + 1$. Therefore, id_cand_i requires $O(n\Delta \log k)$ bits, and the space complexity is also $O(n\Delta \log k)$ bits, which is $O(n\Delta \log n)$ bits by the definition of the parameter k.

Discussion on the parameter k

The maximum expected convergence time and the minimum expected holding time we obtained above are under the assumption on the repetition times $k = \Theta(n)$. Actually, the parameter k here can also be adjusted to make trade-offs between the convergence time and holding time. By Lemma 18, we need $k = \Omega(\log n)$ to get the same result with high probability. If we set k to

be $\Theta(\log n)$, we can get a shorter maximum expected convergence time $O(n \log^2 n)$, while the minimum expected holding time also becomes shorter: $\Omega(poly(n) \cdot \log n)$. Although both times become shorter under the new assumption, we regard the result under the former assumption ($k = \Theta(n)$) has a better performance, since the ratio of the minimum expected holding time to the maximum expected convergence time is exponential in n, while the ratio of the latter one is polynomial.

3.6 Summary

In this chapter, we proposed three systematic approaches to deal with erroneous communications in distributed systems and applied these approaches to design three loosely-stabilizing MIS algorithms and analyze performances of them.

Chapter 4

Loosely-Stabilizing Algorithms on Almost Maximal Independent Set

4.1 Introduction

4.1.1 Background

In many of loose-stabilizing algorithms, including the three algorithms proposed in Chapter 3, an execution period of holding legitimate configurations terminates with a *small* violation of the legitimacy of configurations. In the case of MIS, for example, it is typically terminated by the event that a small number of processes violate the specification of MIS. Conversely, even if the system drops out from legitimate configurations (after a holding period), almost all processes still locally satisfy the specification of MIS. Then one can also expect that the system quickly recovers a legitimate configuration again. Hence the system is expected to keep a large portion of processes "locally" legitimate, for a period much longer than the expected holding time with respect to the specification of MIS. Therefore, we propose the notion of *almost MIS* (ALMIS) to capture such a behavior.

Table 4.1: Performances of the Redundant-State Algorithm on ALMIS and MIS.

	ALMIS			MIS
k_{AL}	$1/\gamma$	$1/o(n)$	$1/\Theta(n)$	0
$maxECT$	$O(\log n)$			
$minEHT$	$\Omega(e^{\Theta(n)})$	$\Omega\left(e^{o(n)}\right)$	$\Omega(poly(n))$	$\Omega(poly(n))$
Space	$O(\log n)$			

4.1.2 Contributions of This Chapter

The concrete results of this chapter are twofold: first, we newly formulate the ALMIS problem fitting our objective. Roughly, our formulation divides the specification of MIS into two properties referred to as *independence* and *maximality*, and quantify the magnitude of violation for each property. The precise definition of ALMIS follows a parametric specification with an acceptable level of violation. The second result is to show that our formulation certainly captures what we want, by demonstrating the loosely-stabilizing redundant-state algorithm presented in Chapter 3 can achieve the exponential minimum expected holding time regarding ALMIS with a reasonably small acceptable level of violation, while it still keeps $O(\log n)$ maximum expected convergence time. In addition, one can also show that the algorithm keeps the precise MIS within most of the holding periods. The performances of the algorithm are summarized in Table 4.1, where k_{AL} is a parameter, n is the number of processes, γ is a sufficiently large constant, $maxECT$ is maximum expected convergence time, and $minEHT$ is minimum expected holding time. The parameter k_{AL} (which will be defined in the latter chapters) represents the extent that the requirements of MIS are relaxed, and by adjusting the value of k_{AL}, we can make a trade-off between the quality of ALMIS and the performance of the algorithm. When $k_{AL} = 0$, the ALMIS problem degenerates to the MIS problem.

4.2 Preliminaries

We assume all the processes are the identical state machine. Let S be the domain of the process states. A configuration of G is a tuple $C = (s_0, s_1, ..., s_{n-1})$ where s_i is the state of process p_i. For a given configuration C and process p, denote by $C(p)$ the state of p in C. Each process can read its state and all of its neighbors' states but can update only its state. Throughout this chapter, we assume the maximum degree among all processes is in constant, i.e., $\Delta = \Theta(1)$. We only consider the uniformly distributed error model M_u (refer to Chapter 3) in this chapter, and we omit the notation of the error model in the following for simplicity.

4.2.1 (f, g)-ALMIS

A set of processes $I \subseteq V$ is called an *independent set* if $\{p, q\} \notin E$ holds for any $p, q \in I$. In addition, if I is not a proper subset of any other independent set, then I is an MIS. This chapter introduces a relaxed notion of MIS, which is called ALMIS. For an MIS I in graph G, we see the independence of I as the property that the cardinality of the edge boundary of I is equal to the summation of the degrees of all processes in I. That is, I is an independent set if and only if it satisfies

$$|\partial(I)| = \sum_{p \in I} \Delta_p.$$

We also see the maximality of I as the property that the number of the processes adjacent to a process in I is equal to the number of the processes not in I. That is, the independent set I is maximal if and only if it satisfies the following equality:

$$|N^*(I)| = |V - I|.$$

Now, we introduce the notion of (f, g)-*ALMIS*, a weaker variant of the standard MIS, by relaxing the two conditions above.

Definition 4.2.1. *Given a distributed system G and non-decreasing functions f and g, the set $I \subseteq V$ is called an (f, g)-ALMIS of G if it satisfies the following two requirements.*

- *f-almost independence:*

$$|\{\{p,q\} \in E \mid p,q \in I\}| \leq f\left(\sum_{p\in I} \Delta_p\right) \tag{4.1}$$

- *g-almost maximality:*

$$|V - I - N^*(I)| \leq g\left(|V - I|\right) \tag{4.2}$$

Intuitively, f-almost independence relaxes the requirement of independence regarding MIS. It allows the cardinality of the edge boundary of I less than the summation of the degree of processes in I with a restricted extent, which is two times the function f of the summation of the degree of processes in I:

$$\sum_{p\in I} \Delta_p - 2f\left(\sum_{p\in I} \Delta_p\right) \leq |\partial(I)|\,.$$

The above inequality implies the existence of edges connecting processes in I, and the number of such edges is not larger than the function f of the summation of the degree of processes in I:

$$|\{\{p,q\} \in E \mid p,q \in I\}| \leq f\left(\sum_{p\in I} \Delta_p\right).$$

Similarly, g-almost maximality relaxes the requirement of maximality regarding MIS. It allows the number of processes adjacent to a process in I less than the number of processes not in I with a restricted extent, which is the function g of the number of processes not in I:

$$|V - I| - g\left(|V - I|\right) \leq |N^*(I)|\,.$$

The above inequality implies the number of processes that are not in I and not adjacent to I is not larger than the function g of the number of processes that are not in I:

$$|V - I - N^*(I)| \leq g\left(|V - I|\right).$$

In this chapter, we consider the case where f and g are linear functions:

$$f(x) = k_1 x \qquad \text{and} \qquad g(x) = k_2 x,$$

where k_1 and k_2 are parameters, and $0 \leq k_1, k_2 \leq 1/\gamma$ where γ is a sufficiently large constant. When $k_1 = k_2 = 0$, ALMIS degenerates to MIS, which is the case we have analyzed in detail in Chapter 3. Therefore, we mainly consider the case of $k_1 + k_2 > 0$ in this chapter, unless stated otherwise.

4.2.2 Redundant-State Algorithm

The redundant-state ($\mathcal{RS}$) algorithm proposed in Chapter 3 is a loosely-stabilizing solution for the MIS problem regarding unreliable communications, which cannot be solved in a self-stabilizing manner. In this chapter, we adopt algorithm $\mathcal{RS}$ for the ALMIS problem with unreliable communications.

The key idea of algorithm $\mathcal{RS}$ is to enlarge the domain of the process state by introducing a large amount of redundancy (more specifically, let $S = \{0, 1, ..., c\}$ where $c = \Theta(n^{d+1})$ and d is a sufficiently large constant) so that even if some obtained state of a neighbor is corrupted by a communication error, it becomes meaningless (i.e., the bit sequence is invalid in the correct behavior of the algorithm) with high probability. In other words, the erroneous state can be detected and does not cause any incorrect effect on the receiver with high probability. This mechanism allows processes to greatly confine the influences from erroneous communications. We adopt Definition 3.3.1 in Chapter 3 and slightly modify it here:

Definition 4.2.2. *We classify processes into four types.*

- *independent process: a process p_i is an independent process if $s_i = c$ and $\forall p_j \in N(p_i) : s_j < c$.*
- *dominated process: a process p_i is a dominated process if $s_i = 0$ and $\exists p_j \in N(p_i) : p_j$ is an independent process.*
- *pseudo-dominated process: a process p_i is a pseudo-dominated process if $s_i = 0$, $\exists p_j \in N(p_i) : s_j = c$, but p_i is not dominated (i.e., there is no independent neighbor).*
- *illegal process: a process p_i is illegal if p_i is not independent, dominated, or pseudo-dominated. That is, (i) $0 < s_i < c$, (ii) $s_i = 0 \wedge \forall p_j \in N(p_i) : s_j < c$, or (iii) $s_i = c \wedge \exists p_j \in N(p_i) : s_j = c$.*

4.3 Specification and Safe Configurations of ALMIS Problem

In this section, we define the specification SP_{AL} of the ALMIS problem and two sets of safe configurations $\mathcal{S}_{AL}$ and $\mathcal{S}_{MIS}$. For convenience, we denote the set of processes with state c in a configuration C by $I(C) = \{p_i \in V \mid C(s_i) = c\}$. We omit C and write just I when no confusion occurs.

Definition 4.3.1. *Given a distributed system G and non-decreasing functions f and g, the specification $SP_{AL}(f, g)$ of ALMIS problem is defined as*

$$SP_{AL}(f, g) \stackrel{\text{def}}{=} (I \text{ is an } (f, g)\text{-ALMIS of } G) \wedge \forall p_i \notin I : s_i = 0$$

Definition 4.3.2. *Given a distributed system G and non-decreasing functions f and g, define $\mathcal{S}_{AL}$ as the set of configurations where the specification $SP_{AL}(f, g)$ is satisfied.*

Define $\mathcal{S}_{MIS}$ as the set of configurations where for any configuration in $\mathcal{S}_{MIS}$, all processes in I are independent and all processes not in I are dominated (or $SP_{AL}(0, 0)$ is satisfied).

From the definition, we have $\mathcal{S}_{MIS} \subseteq \mathcal{S}_{AL}$. The reason for introducing two kinds of safe configurations is that, it is intuitive to introduce $\mathcal{S}_{AL}$ since we are considering the ALMIS problem. However, we will show that $\mathcal{S}_{AL}$ is not a good choice since some configurations in $\mathcal{S}_{AL}$ are very vulnerable regarding the almost maximality requirement, which implies only a constant minimum expected holding time. A smaller set of safe configurations is expected to enable a longer holding time while possibly sacrificing the convergence time. Therefore, we use $\mathcal{S}_{MIS}$ as the set of safe configurations and succeed to achieve the loose-stabilization with exponential holding time and logarithmic convergence time.

4.4 Analysis of Time Complexity

In this section, we analyze the time complexity of algorithm $\mathcal{RS}$ regarding the ALMIS problem. In Section 4.1, we give an overview of the analysis, which shows the analysis for the convergence time is trivial from our previous work, and we cannot achieve loose-stabilization when considering $\mathcal{S}_{AL}$ as safe configurations. In Section 4.2, we show algorithm $\mathcal{RS}$ is a loosely-stabilizing algorithm

regarding ALMIS with an exponentially long minimum expected holding time when considering $\mathcal{S}_{MIS}$ as safe configurations.

4.4.1 Convergence Time and Impossibility Result

We can get the convergence time trivially by Chapter 3. When considering $\mathcal{S}_{MIS}$ as safe configurations, we have the maximum expected convergence time for the ALMIS problem directly by the following lemma.

Lemma 4.4.1 (Theorem 1 in Chapter 3)**.** $\max_{C \in \mathcal{C}} ECT_{\mathcal{RS}}(C, \mathcal{S}_{MIS}) = O(\log n)$ *in terms of steps, where* $\mathcal{C}$ *is the set of all possible configurations of* $\mathcal{RS}$.

Now we consider the case where $\mathcal{S}_{AL}$ is the set of safe configurations. Since $\mathcal{S}_{MIS} \subseteq \mathcal{S}_{AL}$ by Definition 4.3.2, ALMIS can be regarded as an intermediate state of MIS. Thus, the maximum expected convergence time regarding ALMIS is upper-bounded by that regarding MIS, which is $O(\log n)$ steps by Lemma 4.4.1.

Now we show that the above upper bound $O(\log n)$ is tight. To lower-bound the maximum expected convergence time regarding ALMIS, consider the case where initially all processes have states neither 0 nor c. In this case, an ALMIS can be achieved only after all n processes make moves, and we can prove that such a case happens with high probability in $\Theta(\log n)$ steps by the following lemma.

Lemma 4.4.2 (Lemma 1 in Chapter 3)**.** *By taking* $\Theta(\log n)$ *steps from an arbitrary initial configuration, all processes have state 0 or c with probability* $1 - e^{-\Theta(\log n)}$.

Therefore, the maximum expected convergence time regarding ALMIS is lower-bounded by $\Theta(\log n)$. Together with Lemma 4.4.1, we have the following lemma.

Lemma 4.4.3. $\max_{C \in \mathcal{C}} ECT_{\mathcal{RS}}(C, \mathcal{S}_{AL}) = \Theta(\log n)$ *in terms of steps, where* $\mathcal{C}$ *is the set of all possible configurations of* $\mathcal{RS}$.

In the following, we analyze the minimum expected holding time of the case where $\mathcal{S}_{AL}$ is the set of safe configurations, and show loose-stabilization cannot be achieved in this case, as we mentioned in Section 3.1. First, we analyze the bounds of $|I|$ and $|V - I|$ when I is an ALMIS in the following lemma.

Lemma 4.4.4. *If I is an ALMIS, then* $\frac{1}{2(1+\Delta)} \cdot n \leq |I| \leq \frac{2\Delta}{1+2\Delta} \cdot n$.

Proof. First, we have

$$\begin{aligned} n &= |I| + |N^*(I)| + |V - I - N^*(I)| \\ &\leq |I| + \sum_{p \in I} \Delta_p + k_2 |V - I| \\ &\leq (1 + \Delta)|I| + k_2 \cdot (n - |I|) \\ &= (1 + \Delta - k_2)|I| + k_2 n. \end{aligned}$$

We have the second line of inequality by the facts that $|N^*(I)| \leq |\partial(I)|$ and $|\partial(I)| \leq \sum_{p \in I} \Delta_p$. The above inequalities yields

$$|I| \geq \frac{1 - k_2}{1 + \Delta - k_2} \cdot n \geq \frac{1}{2(1 + \Delta)} \cdot n.$$

On the other hand, we have

$$\begin{aligned} |I| &\leq \sum_{p \in I} \Delta_p \\ &= 2\,|\{\{p, q\} \in E \mid p, q \in I\}| + |\partial(I)| \\ &\leq 2\,|\{\{p, q\} \in E \mid p, q \in I\}| + \Delta |N^*(I)| \\ &\leq 2k_1 \sum_{p \in I} \Delta_p + \Delta |N^*(I)| \\ &\leq 2k_1 \Delta |I| + \Delta |N^*(I)|, \end{aligned}$$

which yields

$$|N^*(I)| \geq \frac{1 - 2k_1\Delta}{\Delta} |I| \geq \frac{1}{2\Delta} |I|.$$

Combine the above result and the fact that $|I| + |N^*(I)| \leq n$, we have

$$|I| \leq \frac{2\Delta}{1 + 2\Delta} \cdot n.$$

□

By Lemma 4.4.4, we can directly get

$$\frac{1}{2\Delta} \cdot n \leq |V - I| \leq \frac{1 + 2\Delta}{2 + 2\Delta} \cdot n.$$

After we bound $|I|$ and $|V - I|$, we can also bound the number of illegal processes in the graph when I is an ALMIS in the following lemma. Denote $t^{(\mathrm{ii})}$ and $t^{(\mathrm{iii})}$ as the number of illegal processes satisfying the conditions (ii) or (iii) in the definition of illegal processes (cf. Definition 4.2.2) in the graph, respectively. Notice that we do not consider the illegal process satisfying (i) in the following because all processes have state 0 or c in $\mathcal{S}_{AL}$ by Definition 5, and after that, no processes change its state to values other than 0 or c according to algorithm $\mathcal{RS}$.

Lemma 4.4.5. *If I is an ALMIS and equalities both hold in inequalities (1) and (2), then the lower bounds of $t^{(ii)}$ and $t^{(iii)}$ are $\frac{1}{2\Delta} \cdot k_2 n$ and $\frac{1}{2\Delta(1+\Delta)} \cdot k_1 n$, respectively.*

Proof. The number $t^{(\mathrm{ii})}$ of illegal processes satisfying (ii) is

$$\begin{aligned} t^{(\mathrm{ii})} &= |V - I - N^*(I)| \\ &= k_2|V - I| \\ &\geq \frac{1}{2\Delta} \cdot k_2 n. \end{aligned}$$

The number $t^{(\mathrm{iii})}$ of illegal processes satisfying (iii) is

$$\begin{aligned} t^{(\mathrm{iii})} &\geq \frac{1}{\Delta} \cdot |\{\{p, q\} \in E \mid p, q \in I\}| \\ &= \frac{1}{\Delta} \cdot k_1 \sum_{p \in I} \Delta_p \\ &\geq \frac{1}{\Delta} \cdot k_1 |I| \\ &\geq \frac{1}{2\Delta(1 + \Delta)} \cdot k_1 n. \end{aligned}$$

□

In the following two lemmas, we prove loose-stabilization cannot be achieved when considering $\mathcal{S}_{AL}$ as safe configurations.

Lemma 4.4.6 (Lemma 4 in Chapter 3)**.** *If process p_i is an illegal process, then in any step, the probability that p_i changes its state is $\Theta(1)$.*

Lemma 4.4.7. $\min_{C \in \mathcal{S}_{AL}} EHT_{\mathcal{RS}}(C, SP_{AL}) = \Theta(1)$ *in terms of steps.*

Proof. Consider a configuration $C_i \in \mathcal{S}_{AL}$ where I is an ALMIS and equalities hold in both inequalities (1) and (2). By Lemma 4.4.5, we have

$$t_i^{(\mathrm{ii})} \geq \frac{1}{2\Delta} \cdot k_2 n \quad \text{and} \quad t_i^{(\mathrm{iii})} \geq \frac{1}{2\Delta(1+\Delta)} \cdot k_1 n.$$

Denote $\partial t_i^{(\mathrm{ii})}$ and $\partial t_i^{(\mathrm{iii})}$ as the number of illegal processes satisfying (ii) and (iii) that change state in C_i, respectively. By Lemma 4.4.6, the probability that $\partial t_i^{(\mathrm{ii})} + \partial t_i^{(\mathrm{iii})} > 0$ is (remind Δ is a constant)

$$1 - (1 - \Theta(1))^{t_i^{(\mathrm{ii})} + t_i^{(\mathrm{iii})}} \geq 1 - e^{-(k_1+k_2)\Theta(n)}.$$

Note the fact that when illegal processes satisfying (ii) change states, at most the same number of illegal processes satisfying (iii) will be created (e.g., two adjacent illegal processes satisfying (ii) change states and become illegal processes satisfying (iii) simultaneously); when a single illegal process p satisfying (iii) changes state, at most $\Delta - 1$ illegal processes satisfying (ii) will be created (e.g., $\Delta - 1$ neighbors of p has states 0 and p is the only neighbor of them that had state c). Therefore, after $\partial t_i^{(\mathrm{ii})}$ ($\partial t_i^{(\mathrm{iii})}$, respectively) illegal processes satisfying (ii) ((iii), respectively) change their states, in the worst case we have

$$t_{i+1}^{(\mathrm{ii})} = t_i^{(\mathrm{ii})} - \partial t_i^{(\mathrm{ii})} + (\Delta - 1)\partial t_i^{(\mathrm{iii})}$$

and

$$t_{i+1}^{(\mathrm{iii})} = t_i^{(\mathrm{iii})} - \partial t_i^{(\mathrm{iii})} + \partial t_i^{(\mathrm{ii})}.$$

If $\partial t_i^{(\mathrm{iii})} \geq \partial t_i^{(\mathrm{ii})}$, we have $t_{i+1}^{(\mathrm{ii})} > t_i^{(\mathrm{ii})}$; if $\partial t_i^{(\mathrm{iii})} < \partial t_i^{(\mathrm{ii})}$, we have $t_{i+1}^{(\mathrm{iii})} > t_i^{(\mathrm{iii})}$. If $t^{(\mathrm{ii})}$ increases by 1, the left side of inequality (2) increases by 1 and the right side increases by k_2, which leads to the violation of inequality (2) (remind that equality holds in inequality (2) before $t^{(\mathrm{ii})}$ increases). Similarly, if $t^{(\mathrm{iii})}$ increases by 1, inequality (1) is violated. Therefore, we have $C_{i+1} \notin \mathcal{S}_{AL}$ with probability at least $1 - e^{-(k_1+k_2)\Theta(n)}$, which yields only constant steps of minimum expected holding time (remind $k_1 + k_2 > 0$). □

By Lemmas 4.4.3 and 4.4.7, the minimum expected holding time is much less than the maximum expected convergence time, which yields that loose-stabilization cannot be achieved when considering $\mathcal{S}_{AL}$ as safe configurations. Intuitively, the reason that causes the impossibility is the possible existence of pseudo-dominated processes that can potentially become illegal

processes, as we have shown in the proof of Lemma 4.4.7. In the next section, we focus on the analysis of the holding time when considering $\mathcal{S}_{MIS}$ as safe configurations, and loose-stabilization can be achieved in this case.

4.4.2 Holding Time

In this section, we show that algorithm $\mathcal{RS}$ has an exponentially long minimum expected holding time when considering $\mathcal{S}_{MIS}$ as safe configurations, which yields that $\mathcal{RS}$ can achieve loose-stabilization for the ALMIS problem. In any execution starting from a safe configuration in $\mathcal{S}_{MIS}$, no process changes its state to a value other than 0 or c. Thus, we do not need to consider the illegal process satisfying (i) in the following.

Denote PI as the total number of pseudo-dominated and illegal processes. We analyze how the action of each process affects the graph in the following two lemmas. More specifically, in the following lemmas, we prove that if independent or dominated processes change states, PI will increase; if pseudo-dominated or illegal processes change states, PI does not increase.

Lemma 4.4.8. *If an independent or dominated process changes its state, PI increases by at most $\Delta^2 + 1$.*

Proof. Let p be the process that changes its state. When p is an independent process and changes its state to 0, it becomes an illegal process satisfying (ii). A neighbor q of p also becomes an illegal process satisfying (ii) when p was the only independent process adjacent to q. Therefore, PI increases by at most $\Delta + 1$.

When p is a dominated process and changes its state to c, it becomes an illegal process satisfying (iii). A neighbor q of p that was independent also becomes an illegal process satisfying (iii), and all neighbors of q except p becomes pseudo-dominated processes when q was the only independent process adjacent to them. Therefore, PI increases by at most $\Delta^2 + 1$. □

Lemma 4.4.9. *If pseudo-dominated or illegal processes change states, PI does not increase.*

Proof. By Definition 4.2.2, independent processes are not adjacent to any pseudo-dominated or illegal process (remind that we do not consider illegal processes satisfying (i) here, since no process changes its state to a value other than 0 or c). Therefore, independent processes are not affected

and remain independent when pseudo-dominated and illegal processes change their states. On the other hand, dominated processes may be adjacent to pseudo-dominated and illegal processes. However, since dominated processes are adjacent to independent processes by Definition 4.2.2 and independent processes remain independent when pseudo-dominated and illegal processes change their states, dominated processes also remain dominated when pseudo-dominated and illegal processes change their states. Therefore, PI does not increase. □

By Lemma 4.4.5, we know that violation of $SP_{AL}(f, g)$ requires $t^{(\mathrm{ii})} > \frac{1}{2\Delta} \cdot k_2 n$ or $t^{(\mathrm{iii})} > \frac{1}{2\Delta(1+\Delta)} \cdot k_1 n$. Denote $k_{AL} = \min\{\frac{1}{2\Delta(1+\Delta)} \cdot k_1, \frac{1}{2\Delta} \cdot k_2\}$ for simplicity. In the following, we analyze the case

$$PI > k_{AL} n$$

that is weaker than the previous one so that we can upper-bound the probability that the previous case happens. To do this, we use the results of the following lemmas from Chapter 3.

Lemma 4.4.10 (Lemma 2 in Chapter 3)**.** *If process p_i is an independent, dominated or pseudo-dominated process, then in any step, the probability that p_i keeps the same state is $1 - O(1/c)$, where $c = \Theta(n^{d+1})$ and d is a sufficiently large constant.*

Lemma 4.4.11 (Lemma 5 in Chapter 3)**.** *If process p_i is an illegal process, then in any step, the probability that p_i becomes independent is $\Omega(1)$.*

Lemma 4.4.12 (Lemma 6 in Chapter 3)**.** *If process p_i is a pseudo-dominated process, then in any step, the probability that p_i becomes dominated is $\Omega(1)$.*

Denote P_l as the lower bound of the probabilities that an illegal process becomes independent and a pseudo-dominated process becomes dominated in any step. By Lemmas 4.4.11 and 4.4.12, P_l is a constant greater than 0.

Lemma 4.4.13. *Given an initial configuration $C_0 \in \mathcal{S}_{MIS}$ and an execution $E \in E_{\mathcal{RS}}(C_0)$. In any configuration C_i in E where $i \geq 0$, we have $PI_i \leq k_{AL} n$ with probability at least $1 - e^{-\Theta(k_{AL} \cdot n)}$, where PI_i is the value of PI in C_i.*

Proof. We prove the lemma by proving the following claims.

Claim 1. *If $PI_i = o(k_{AL}n)$, then $PI_{i+1} \leq k_{AL}n$ with probability at least $1 - O(n^{-\Theta(k_{AL}\cdot n)})$.*

Proof. By Lemma 4.4.8, PI_i increases by at most $\Delta^2 + 1$ if an independent or dominated process changes its state. Therefore, if at most $\frac{1}{2(\Delta^2+1)} \cdot k_{AL}n$ independent and dominated processes change states, we have

$$PI_{i+1} \leq PI_i + (\Delta^2 + 1) \cdot \frac{1}{2(\Delta^2 + 1)} \cdot k_{AL}n < k_{AL}n.$$

The probability such a case happens is (remind Δ is a constant)

$$1 - \binom{n - PI_i}{\frac{1}{2(\Delta^2+1)} \cdot k_{AL}n} \cdot O\left(\frac{1}{c}\right)^{\frac{1}{2(\Delta^2+1)} \cdot k_{AL}n} \geq 1 - O(n^{-\Theta(k_{AL}\cdot n)}).$$

□

Claim 2. *If $PI_i = \tau \cdot k_{AL}n$ for some constant τ s.t. $0 < \tau \leq 1$ in C_i, then $PI_{i+1} < PI_i$ with probability at least $1 - e^{-\Theta(k_{AL}\cdot n)}$.*

Proof. By Lemma 4.4.9, PI_i does not increase if pseudo-dominated or illegal processes change states. Moreover, pseudo-dominated and illegal processes may become dominated or independent with probability at least P_l (remind P_l is a constant) by Lemmas 4.4.11 and 4.4.12. By Hoeffding's inequality [45], the probability that at least $\frac{1}{2}P_l \cdot PI_i$ illegal processes independent in one step is at least

$$1 - e^{-2PI_i \cdot \left(P_l - \frac{\frac{P_l}{2} \cdot PI_i}{PI_i}\right)^2} = 1 - e^{-\Theta(k_{AL}n)}.$$

By a similar analysis with Claim 1, the probability that $\frac{P_l}{4} \cdot PI_i$ independent and dominated processes change states in one step is at least $1 - O(n^{-\Theta(k_{AL}\cdot n)})$. Therefore, with probability

$$(1 - e^{-\Theta(k_{AL}n)}) \cdot (1 - O(n^{-\Theta(k_{AL}\cdot n)})) = 1 - e^{-\Theta(k_{AL}n)},$$

we have

$$PI_{i+1} \leq PI_i - \frac{1}{2}P_l \cdot PI_i + \frac{1}{4}P_l \cdot PI_i < PI_i.$$

□

Initially, we have $PI_0 = 0$ since $C_0 \in \mathcal{S}_{MIS}$. At each configuration C_i ($i > 0$) after C_0, if PI_{i-1} is $o(k_{AL}n)$, e.g., $PI_{i-1} = 0$, we have $PI_i \leq k_{AL}n$ with high probability by Claim 1; if $PI_{i-1} = \tau \cdot k_{AL}n$ for some constant τ s.t. $0 < \tau \leq 1$, e.g., $PI_{i-1} = \frac{1}{2} \cdot k_{AL}n$ or $PI_{i-1} = k_{AL}n$, we have $PI_i < PI_{i-1} \leq k_{AL}n$ with high probability by Claim 2. Therefore, we have the lemma. □

Combining the result of Chapter 3 and Lemma 4.4.13, we can analyze the holding time in the following lemma.

Lemma 4.4.14.

$$\min_{C \in \mathcal{S}_{MIS}} EHT_{\mathcal{RS}}(C, SP_{AL}) = \Omega\left(poly(n) + e^{\Theta(k_{AL} \cdot n)}\right)$$

in terms of steps.

Proof. To deviate from the specification of ALMIS in executions with initial configurations in $\mathcal{S}_{MIS}$, the system should deviate from $\mathcal{S}_{MIS}$ and also reach a configuration with $PI > k_{AL}n$. By the result of Chapter 3, the minimum expected number of steps to deviate from $\mathcal{S}_{MIS}$ is $\Omega(poly(n))$.

By Lemma 4.4.13, we have $PI \leq k_{AL}n$ with probability at least $1 - e^{-\Theta(k_{AL}n)}$ for any configuration in execution starting from an initial configuration in $\mathcal{S}_{MIS}$. Hence, the probability that $PI > k_{AL}n$ is at most $e^{-\Theta(k_{AL}n)}$, which yields the minimum expected $1/e^{-\Theta(k_{AL}n)} = e^{\Theta(k_{AL}n)}$ steps up to the first configuration satisfying $PI > k_{AL}n$, assuming the execution starts from a configuration in $\mathcal{S}_{MIS}$. Therefore, we can obtain the minimum expected holding time of $\Omega\left(\max\{poly(n), e^{\Theta(k_{AL} \cdot n)}\}\right) = \Omega\left(poly(n) + e^{\Theta(k_{AL} \cdot n)}\right)$. □

Combining Lemmas 4.4.1 and 4.4.14, we have our main theorem of the chapter.

Theorem 4.4.1. *Algorithm $\mathcal{RS}$ is a $\left(O(\log n), \Omega\left(poly(n) + e^{\Theta(k_{AL} \cdot n)}\right)\right)$-loosely-stabilizing algorithm regarding the ALMIS problem.*

By choosing different k_{AL}, we can leverage the quality of ALMIS and its minimum expected holding time.

- When $k_{AL} = 1/\gamma$ where γ is a sufficiently large constant, the minimum expected holding time is $\Omega(e^{\Theta(n)})$.
- When $k_{AL} = 1/o(n)$, e.g., $k_{AL} = 1/\log n$, the minimum expected holding time is $\Omega(e^{\Theta(n/\log n)})$, which is sub-exponentially long.

- When $k_{AL} = \Theta(1/n)$ or 0, the minimum expected holding time is $\Omega(poly(n))$, which is polynomial long. Note that when $k_{AL} = 0$, the problem degenerates to the MIS problem.

4.5 Summary

In this chapter, we formulated the definition of almost MIS (ALMIS) and applied the loosely-stabilizing redundant-state algorithm regarding ALMIS. We showed that by considering ALMIS, the algorithm can keep the logarithmic maximum expected convergence time while achieving at most the exponential minimum expected holding time by the choice of quality of ALMIS.

Chapter 5

Conclusion

5.1 Summary of the Results

In this dissertation, we consider loosely-stabilizing MIS algorithms adapting to probabilistic erroneous communications.

In Chapter 3, we considered self-stabilization tolerant to erroneous communications and applied the concept of loose-stabilization to circumvent the impossibility of closure property in networks with probabilistically erroneous communication. Specifically, we presented three systematic approaches for designing loose-stabilizing algorithms in probabilistic error models: the redundant-state, the step-up, and the repetition approaches. We considered two different models of erroneous communication: the uniformly distributed error and the arbitrarily distributed error, and analyzed the performances of three approaches. The redundant-state approach assumes a constant maximum degree and uniformly distributed error model with maximum expected convergence time of $O(\log n)$, minimum expected holding time $\Omega(n^d)$, and space complexity $O(\log n)$. The step-up approach only assumes the uniformly distributed error model with maximum expected convergence time of $O(n \log^3 n)$, minimum expected holding time of $\Omega(n^d)$, and space complexity $O(\log n)$. The repetition approach assumes a unique identifier for each process and the arbitrarily distributed error model with maximum expected convergence time of $O(n^2 \log n)$, minimum expected holding time of $e^{\Omega(n)}$, and space complexity $O(n\Delta \log n)$.

In Chapter 4, we considered a relaxed notion of the MIS problem, named ALMIS, and

formulated its definition. We also showed that the loosely-stabilizing redundant-state algorithm presented in Chapter 3 can achieve the exponential minimum expected holding time regarding ALMIS, while still keeping $O(\log n)$ maximum expected convergence time and space complexity. By adjusting the parameter k_{AL}, which represents the extent that the requirements of MIS are relaxed, we can make a trade-off between the quality of ALMIS and the performance of the algorithm. When $k_{AL} = 0$, the ALMIS problem degenerates to the MIS problem.

5.2 Future Directions

The technique of loose-stabilization is not widely used for now. The concept was originally defined in the probabilistic population protocol (PPP) model and was deeply investigated in the model. To our knowledge, besides this dissertation, loose-stabilization is only used in the leader election problem in the PPP model and the congestion control problem in the message passing model. The common point of these works is that their computational models all assume the communications or interactions between processes in the system follow a probabilistic way. For the PPP model used in [39, 35, 37, 38, 40], it is assumed that the interaction between any two processes happens in a uniform probability. For the message passing model in [42], the network consists of multiple clients and a common fixed server where all clients know the server, and each client sends a message to the server with some probability. In this dissertation, we assume an atomic-state model with unreliable communications where each process may read the incorrect value from its neighbors with a constant probability. When such "probabilistic interaction" happens, loose-stabilization is a more promising approach than self-stabilization since the safe configurations always have the potential to become illegitimate.

Although many problems like leader elections have self-stabilizing solutions, if we alter the ideal setting and introduce some probabilistic faults, such as unreliable communications, self-stabilizing solutions may become impossible to achieve. At this moment, we can try to apply loose-stabilization and get a solution that may not be perfect but is totally enough for practical use.

The error-correction and error-detection mechanisms employed in this dissertation are not exhaustive within the context of loose-stabilization. However, it is possible to enhance the

development of loosely-stabilizing algorithms by designing suitable hash functions or utilizing alternative error-correction and error-detection schemes. These modifications can cater to different assumptions and accommodate various circumstances.

For example, when dealing with specific network topologies like circles, grids, and cliques, loosely-stabilizing algorithms may require adjustments to consider the unique characteristics of each topology. Additionally, in real-world scenarios, distributed systems often face resource limitations such as restricted bandwidth, power constraints, or memory restrictions. By adapting loose-stabilization algorithms, it becomes possible to optimize resource utilization and effectively address the specific constraints imposed by the system.

By incorporating tailored error-correction/detection mechanisms and making appropriate adjustments, loosely-stabilizing algorithms can be designed to adapt to a wide range of circumstances and effectively operate in diverse environments.

Acknowledgments

I am deeply grateful for the invaluable support and guidance I have received throughout the journey of completing my doctorate degree. This accomplishment would not have been possible without the contributions and encouragement of numerous individuals and institutions.

First and foremost, I would like to express my deepest appreciation to my supervisor Professor Toshimitsu Masuzawa for his kind guidance and encouragement. I also would like to extend my gratitude to Professor Fumihiko Ino, Professor Yoshiki Higo, Associate Professor Taisuke Izumi and Assistant Professor Naoki Kitamura of Graduate School of Information Science and Technology, Osaka University, and Associate Professor Yuichi Sudo of Hosei University for their precious advice and valuable comments on my research and dissertation.

I would like to thank the professors and staff of Osaka University Inter-Disciplinary Fellowship for their financial support. I also would like to thank Ms. Noriko Sato for her kind support. I am also grateful to the staff and students of Algorithm Engineering Laboratory, Graduate School of Information Science and Technology, Osaka University.

Finally, I would like to thank my family for their kindness and support during my life at the university.

Bibliography

[1] Edsger W Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM, 17 (11): 643-644, 1974*.

[2] Sébastien Tixeuil. *Vers l'auto-stabilisation des systèmes à grande échelle*. PhD thesis, Université Paris Sud-Paris XI, 2006.

[3] Gerard Tel. *Introduction to distributed algorithms*. Cambridge university press, 2000.

[4] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986.

[5] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *Journal of the ACM (JACM)*, 63(3):1–45, 2016.

[6] Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 270–277. SIAM, 2016.

[7] Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 350–363, 2020.

[8] Mohsen Ghaffari, Christoph Grunau, and Václav Rozhoň. Improved deterministic network decomposition. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2904–2923. SIAM, 2021.

[9] Salwa Faour, Mohsen Ghaffari, Christoph Grunau, Fabian Kuhn, and Václav Rozhoň. Local distributed rounding: Generalized to mis, matching, set cover, and beyond. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4409–4447. SIAM, 2023.

[10] Guy E Blelloch, Richard Peng, and Kanat Tangwongsan. Linear-work greedy parallel approximate set cover and variants. In *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*, pages 23–32, 2011.

[11] Mohsen Ghaffari. Distributed maximal independent set using small messages. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 805–820. SIAM, 2019.

[12] Nathan Linial. Distributive graph algorithms global solutions from local data. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 331–335. IEEE, 1987.

[13] Weili Wu, Hongwei Du, Xiaohua Jia, Yingshu Li, and Scott C-H Huang. Minimum connected dominating sets and maximal independent sets in unit disk graphs. *Theoretical Computer Science*, 352(1-3):1–7, 2006.

[14] SC-H Huang, P-J Wan, Chinh T Vu, Yingshu Li, and Frances Yao. Nearly constant approximation for data aggregation scheduling in wireless sensor networks. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, pages 366–372. IEEE, 2007.

[15] Ozkan Arapoglu, Vahid Khalilpour Akram, and Orhan Dagdeviren. An energy-efficient, self-stabilizing and distributed algorithm for maximal independent set construction in wireless sensor networks. *Computer Standards & Interfaces*, 62:32–42, 2019.

[16] Alper Köse and Berna Özbek. Resource allocation for underlaying device-to-device communications using maximal independent sets and knapsack algorithm. In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–5. IEEE, 2018.

[17] Robert A Murphey, Panos M Pardalos, and Mauricio GC Resende. Frequency assignment problems. *Handbook of Combinatorial Optimization: Supplement Volume A*, pages 295–377, 1999.

[18] Vinod Maan and GN Purohit. A distributed approach for frequency allocation using graph coloring in mobile networks. *International Journal of Computer Applications*, 58(6), 2012.

[19] VP Shilo. New lower bounds of the size of error-correcting codes for the z-channel. *Cybernetics and Systems Analysis*, 38(1):13, 2002.

[20] Noboru Hamada and Fumikazu Tamari. On a geometrical method of construction of maximal t-linearly independent sets. *Journal of Combinatorial Theory, Series A*, 25(1):14–28, 1978.

[21] Walter Whiteley. Counting out to the flexibility of molecules. *Physical Biology*, 2(4):S116, 2005.

[22] Steffen Klamt, Utz-Uwe Haus, and Fabian Theis. Hypergraphs and cellular networks. *PLoS computational biology*, 5(5):e1000385, 2009.

[23] Saket Navlakha and Ziv Bar-Joseph. Algorithms in nature: the convergence of systems biology and computational thinking. *Molecular systems biology*, 7(1):546, 2011.

[24] Sandra M Hedetniemi, Stephen T Hedetniemi, David P Jacobs, and Pradip K Srimani. Self-stabilizing algorithms for minimal dominating sets and maximal independent sets. *Computers & Mathematics with Applications*, 46(5-6):805–811, 2003.

[25] Volker Turau. Making randomized algorithms self-stabilizing. In *International Colloquium on Structural Information and Communication Complexity*, pages 309–324. Springer, 2019.

[26] Yehuda Afek and Geoffrey M Brown. Self-stabilization over unreliable communication media. *Distributed Computing*, 7(1):27–34, 1993.

[27] Shlomi Dolev, Swan Dubois, Maria Potop-Butucaru, and Sébastien Tixeuil. Stabilizing data-link over non-fifo channels with optimal fault-resilience. *Information Processing Letters*, 111(18):912–920, 2011.

[28] Amos Israeli and Marc Jalfon. Token management schemes and random walks yield self-stabilizing mutual exclusion. In *Proceedings of the ninth annual ACM symposium on Principles of distributed computing*, pages 119–131, 1990.

[29] Mohamed G Gouda. The theory of weak stabilization. In *International Workshop on Self-Stabilizing Systems*, pages 114–123. Springer, 2001.

[30] Ji-Cherng Lin, Tetz C Huang, Cheng-Zen Yang, and Nathan Mou. Quasi-self-stabilization of a distributed system assuming read/write atomicity. *Computers & Mathematics with Applications*, 57(2):184–194, 2009.

[31] Stéphane Devismes, Sébastien Tixeuil, and Masafumi Yamashita. Weak vs. self vs. probabilistic stabilization. In *2008 The 28th International Conference on Distributed Computing Systems*, pages 681–688. IEEE, 2008.

[32] S Tixeuil. Algorithms and theory of computation handbook. *Self-Stabilizing Algorithms, second edition, in: Chapman & Hall/CRC Appl. Algorithms Data Struct. Ser., CRC Press, Taylor & Francis Group*, pages 26–1, 2009.

[33] Joffroy Beauquier, Christophe Genolini, and Shay Kutten. k-stabilization of reactive tasks. In *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, page 318, 1998.

[34] Christophe Genolini and Sébastien Tixeuil. A lower bound on dynamic k-stabilization in asynchronous systems. In *21st IEEE Symposium on Reliable Distributed Systems, 2002. Proceedings.*, pages 212–221. IEEE, 2002.

[35] Yuichi Sudo, Junya Nakamura, Yukiko Yamauchi, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Loosely-stabilizing leader election in a population protocol model. *Theoretical Computer Science*, 444:100–112, 2012.

[36] Dana Angluin, James Aspnes, Michael J Fischer, and Hong Jiang. Self-stabilizing population protocols. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(4):1–28, 2008.

[37] Yuichi Sudo, Fukuhito Ooshita, Hirotsugu Kakugawa, Toshimitsu Masuzawa, Ajoy K Datta, and Lawrence L Larmore. Loosely-stabilizing leader election for arbitrary graphs in population protocol model. *IEEE Transactions on Parallel and Distributed Systems*, 30(6):1359–1373, 2018.

[38] Yuichi Sudo, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Loosely stabilizing leader election on arbitrary graphs in population protocols without identifiers or random numbers. *IEICE Transactions on Information and Systems*, 103(3):489–499, 2020.

[39] Taisuke Izumi. On space and time complexity of loosely-stabilizing leader election. In *International Colloquium on Structural Information and Communication Complexity*, pages 299–312. Springer, 2015.

[40] Yuichi Sudo, Fukuhito Ooshita, Hirotsugu Kakugawa, Toshimitsu Masuzawa, Ajoy K Datta, and Lawrence L Larmore. Loosely-stabilizing leader election with polylogarithmic convergence time. *Theoretical Computer Science*, 806:617–631, 2020.

[41] Yuichi Sudo, Ryota Eguchi, Taisuke Izumi, and Toshimitsu Masuzawa. Time-Optimal Loosely-Stabilizing Leader Election in Population Protocols. In Seth Gilbert, editor, *35th International Symposium on Distributed Computing (DISC 2021)*, volume 209 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 40:1–40:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[42] Michael Feldmann, Thorsten Götte, and Christian Scheideler. A loosely self-stabilizing protocol for randomized congestion control with logarithmic memory. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pages 149–164. Springer, 2019.

[43] Ran Tamir, Ariel Livshits, and Yonatan Shadmi. Simple majority consensus in networks with unreliable communication. *Entropy*, 24(3):333, 2022.

[44] Mahyar Shirvanimoghaddam, Ayoob Salari, Yifeng Gao, and Aradhika Guha. Federated learning with erroneous communication links. *IEEE Communications Letters*, 26(6):1293–1297, 2022.

[45] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994.