Doctoral Dissertation

A Study on Efficient Algorithms for Local Interaction Systems

局所的な相互作用系のための効率的なアルゴリズムに関する研究

Naoki Kitamura

Department of Computer Science and Engineering,
Nagoya Institute of Technology, Nagoya, Japan

Supervisor: Professor Yoshiaki Katayama

2022

Abstract

The local interaction system is the system in which multiple computing entities communicate locally to achieve a specific goal, and many models of the local interaction system has been proposed in relation to a vast class of real complex systems. One of the primary interests in the context of theoretical computer science is to reveal the computational power of a given local interaction systems. This dissertation investigates the capability of two models respectively derived from distributed systems of networked computers and mathematical puzzles.

In the theory of distributed computing, the *CONGEST* model is a standard computational model for distributed graph algorithm. A distributed system is represented by a simple undirected connected graph $G = (V(G), E(G))$. Let n and m be the numbers of nodes and edges, respectively. The CONGEST model is defined as a round based synchronous system with bandwidth, where each link can transfer a small message of $O(\log n)$ bits per round. The limited bandwidth in the CONGEST model precludes a trivial universal solution for every graph problem, where the leader node collects all the topological information of G and solves the problem using a centralized algorithm. This approach takes $O(n^2)$ rounds in the worst case of $m = \Omega(n^2)$. The technical challenge in designing CONGEST algorithms concerns how each node computes a fragment of the solution without information on the whole input instance. The local interaction system is also studied in the field of mathematical puzzle. Many of mathematical puzzles such as *15-puzzles* and *sokoban* can be seen as a *transformation problem* of a local interaction systems. Transformation problems can be formulated as "Can we transform configuration A into configuration B, if certain transformations only are allowed? ". Unfortunately, many problems related to the transformation problem of a puzzle are known to be NP-hard or PSPACE-complete. Therefore, in the context of the transformation problems, the existence of an algorithm that solves the transformation problem in polynomial time is one criteria of efficiency.

In this dissertation, we construct the efficient algorithm for local interaction system. In Chapter 2, we consider the minimum spanning tree (MST) problem in the CONGEST model. It is known that the lower bound of the MST is $\tilde{\Omega}(\sqrt{n}+D)$ rounds in the CONGEST model. This lower bounds is derived from special "hard-core" instances, and do not necessarily apply to popular graph classes such as planar graphs, which evokes the interest of developing efficient distributed graph algorithms for specific graph classes. In this dissertation, we study the relationship between several major graph parameters and the running time of MST. In particular, we focus on the following three parameters, that is: (1) doubling dimension, (2) diameter, and (3) clique width. As a positive result, we show that there is an $\tilde{O}(D^x)$-round algorithm that constructs a MST for any doubling dimension-x graph. We show that there exists an algorithm for constructing a MST in $\tilde{O}(n^{1/4})$ rounds for any graph of diameter three. In addition, we show that there exists an algorithm for constructing a MST in $\tilde{O}(n^{1/3})$ rounds for any graph of diameter four. These results are similar to the long-standing complexity gap of the MST construction in graphs with small diameters, which was originally proposed by Lotker et al. We present a negative instance certifying that bounded clique-width does not help in the construction of a MST. More precisely, we provide an instance of clique-width six, where the construction of MST is as expensive as the general case, i.e., $\tilde{\Omega}(\sqrt{n}+D)$ rounds.

In Chapter 3, we consider the maximum matching algorithm in the CONGEST model. Although local algorithms for the approximate maximum matching problem have been widely studied, exact algorithms have not been much studied. In fact, no exact maximum matching algorithm that is faster than the trivial upper bound of $O(n^2)$ rounds is known for general instances. In this

dissertation, we proposed a randomized $O(s_{\max}^{3/2})$-round algorithm in the CONGEST model, where $s_{\max}$ is the size of maximum matching. This is the first exact maximum matching algorithm in $o(n^2)$ rounds for general instances in the CONGEST model.

In Chpater 4, we treat the mathematical pachinko problems as a kind of the transformation problem. Recently, several mathematical models of Pachinko have been proposed. A number of pins are spiked in a field. A ball drops from the top of the playfield and the ball falls down. In the 50-50 model, if the ball hits a pin, it moves to the left or right passage of the pin with an equal probability. An arrangement of pins generates a distribution of the drop probability for all of the columns. This problem was considered by generating uniform distributions. Previous studies have demonstrated that the $(1/2^a)$-uniform distribution is possible for $a \in \{0, 1, 2, 3, 4\}$ and is conjectured so that it is possible for any positive integer a. We describe the constructive proof for this conjecture. In other words, for any $a \geq 1$, we provide the pin arrangement that generates the $(1/2^a)$-uniform distribution. This construction consumes $O(2^{3a})$ pins for generating the $(1/2^a)$-uniform distribution. We also formalizes a natural decision problem yielded by this model while investigating its computational complexity. More precisely, given any drop-probability distribution A and any partial drop-probability distribution B, we show that it is non-deterministic polynomial-time (NP) hardness to determine if there exists a pin arrangement that transforms A into B.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

The local interaction system is a system in which multiple computing entities communicate locally to achieve a specific goal, and many models of the local interaction system has been proposed, which is not limited to the field of computer science: computer network, swarm of robots or wild animals, chemical reaction, cell interaction in human bodies, and mathematical puzzles. One of the primary interests in the context of theoretical computer science is to reveal the computational power of a given local interaction systems. This dissertation investigates two models respectively derived from distributed systems of networked computers and mathematical puzzles.

In the theory of distributed computing, the *CONGEST* model is a standard computational model for distributed graph algorithm. A distributed system is represented by a simple undirected connected graph $G = (V(G), E(G))$. Let n and m be the numbers of nodes and edges, respectively. The CONGEST model is defined as a round based synchronous system with bandwidth, where each link can transfer a small message of $O(\log n)$ bits per round. The limited bandwidth in the CONGEST model precludes a trivial universal solution for every graph problem, where the leader node collects all the topological information of G and solves the problem using a centralized algorithm. This approach takes $O(n^2)$ rounds in the worst case of $m = \Omega(n^2)$. The technical challenge in designing CONGEST algorithms concerns how each node computes a fragment of the solution without information on the whole input instance. The recent development of design techniques for CONGEST algorithms has yielded many efficient solutions for various graph problems such as the minimum spanning tree [37, 44, 49, 55, 57, 63], distance problems including shortest-path computation [13, 30, 43, 50, 52, 64, 70], and flow and cut [20, 24, 38, 39, 71]. Owing to the existence of the $O(n^2)$-round universal algorithm, the weakest non-trivial challenge in the design of a CONGEST algorithms is to achieve a subquadratic $o(n^2)$-round upper bound. In contrast to the universal upper bound, all the problems listed above belong to the class of *global* problems exhibiting an $\Omega(D)$-round lower bound, where D is the diameter of the input graph G. Thus, the tight round complexities of global problems lie between $\Theta(n^2)$ and $\Theta(D)$. For many of global problems, near-tight complexity bounds, typically $\tilde{\Theta}(\sqrt{n} + D)$ rounds or $\tilde{\Theta}(n)$ rounds, have been proved [10, 32, 76].

The local interaction system is also studied in the field of mathematical puzzle. For example, in 15-puzzles, the system is defined as a model in which 16 squares are regarded as entities and each square can communicate (i.e. delivery of pieces) only with adjacent squares. Many of mathematical puzzles such as 15-puzzles can be seen as a *transformation problem* of a local interaction systems [22, 46]. Transformation problems can be formulated as "Can we transform configuration A into configuration B, if certain transformations only are allowed?". Unfortunately, many problems

related to the transformation problem are known to be NP-hard or PSPACE-complete. Therefore, in the context of the transformation problem, the existence of an algorithm that solves the transformation problem in polynomial time is one criteria of efficiency.

.

1.2 Overview of This Dissertation

In this dissertation, we investigate the design of distributed algorithms in the CONGEST model (In Chapter 2 and Chapter 3), and the transformation problem for the mathematical model of pachinko (In Chpater 4).

1.2.1 Efficient CONGEST Algorithms for Restricted Graph Classes

In Chapter 2, we consider the *minimum spanning tree* (MST) problem in the CONGEST model. Given a graph with edge weights, the MST problem is the problem of constructing a spanning tree with the smallest sum of edge weights. It is known that the lower bound of the MST problem is $\tilde{\Omega}(\sqrt{n}+D)$ rounds in the CONGEST model [63] (D is the diameter of the graph). The $\tilde{\Omega}(\sqrt{n}+D)$-rounds lower bound is derived from special "hard-core" instances, and do not necessarily apply to popular graph classes such as planar graphs, which evokes the interest of developing efficient distributed graph algorithms for specific graph classes. In the last few years, the study along this line has rapidly made progress, where the concepts of *partwise aggregation* and *low-congestion shortcut* play an important role. In the partwise aggregation problem, all nodes in the network are initially partitioned into a number of disjoint-connected subgraphs known as a *part*. The goal of this problem is to perform a certain type of distributed task independently within all the parts in parallel. The executable tasks cover several standard operations, such as broadcast, convergecast, leader election, and finding minimum. The low-congestion shortcut is a framework for solving the partwise aggregation problem, which is initiated by Ghaffari and Haeupler [37]. The key difficulty of the partwise aggregation problem appears when the diameter of a part is much larger than the diameter D of the original graph. Because the diameter of a part can become $\Omega(n)$ in the worst case scenario, the naive solution that performs the aggregation task only by in-part communication causes the expensive $\Omega(n)$-round running time. A low-congestion shortcut is defined as the sets of links augmented to each part to accelerate the aggregation task there. Its efficiency is characterized by the following two quality parameters: The *dilation* is the maximum diameter of all the parts after the augmentation, whereas the *congestion* is the maximum edge congestion of all edges e, where the edge congestion of e is defined as the number of parts augmenting e. In the application of low-congestion shortcuts, the performance of an algorithm typically relies on the sum of the dilation and congestion. Hence, we simply refer to the value of dilation plus congestion as the *quality* of the shortcut. It is known that any low-congestion shortcut with quality q and $O(f)$-round construction time yields an $\tilde{O}(f+q)$-round solution for the partwise aggregation problem, and $\tilde{O}(f+q)$-round partwise aggregation yields the efficient solutions for several fundamental graph problems including the MST problem. Precisely, the following meta-theorem holds:

Theorem 1 (Ghaffari and Haeupler [37], Haeupler and Li [50])**.** *Let $\mathcal{G}$ be a graph class allowing the low-congestion shortcut with quality $O(q)$ that can be constructed in $O(f)$ rounds in the CONGEST model. Then, there exist three algorithms solving (1) the MST problem in $\tilde{O}(f+q)$ rounds, (2) the $(1+\epsilon)$-approximate minimum cut problem in $\tilde{O}(f+q)$ rounds for any $\epsilon = \Omega(1)$, and (3) $O(n^{O(\log\log n)/\log\beta})$-approximate weighted single-source shortest path problem in $\tilde{\Omega}((f+q)\beta)$ rounds*

for any $\beta = \Omega(\text{polylog}(n))$[1].

Conversely, if we obtain a time-complexity lower bound for any problem stated above, then it also applies to the partwise aggregation and low-congestion shortcut (with respect to quality plus construction time). In fact, the $\tilde{\Omega}(\sqrt{n} + D)$-round lower bound of shortcuts for general graphs is deduced from the lower bound of MST. Meanwhile, the existence of efficient (in the sense of breaking the general lower bound) low-congestion shortcut is known for several major graph classes as well as its construction algorithms [37, 42, 43, 48, 49, 51].

We study the relationship between several major graph parameters and the quality of low-congestion shortcut. In particular, we focus on the following four parameters, that is: (1) doubling dimension, (2) diameter, and (3) clique width. The precise statement of our results is as follows:

- There is an $O(1)$-round algorithm that constructs a low congestion shortcut with quality $\tilde{O}(D^x)$ for any doubling dimension-x graph.
- There exists an algorithm for constructing a low-congestion shortcut with quality $\tilde{O}(n^{1/4})$ in $\tilde{O}(n^{1/4})$ rounds for any graph of diameter three. In addition, there exists an algorithm for constructing a low-congestion shortcut with quality $\tilde{O}(n^{1/3})$ in $\tilde{O}(n^{1/3})$ rounds for any graph of diameter four. These results are similar to the long-standing complexity gap of the MST construction in graphs with small diameters, which was originally proposed by Lotker et al. [67].
- We present a negative instance certifying that bounded clique-width does not help in the construction of good-quality shortcuts. More precisely, we provide an instance of clique-width six, where the construction of MST is as expensive as the general case, i.e., $\tilde{\Omega}(\sqrt{n}+D)$ rounds.

Table 1.1 summarizes the state-of-the-art upper and lower bounds for low-congestion shortcut. Notably, all the parameters considered in this study are independent of the other (known) parameters admitting good shortcuts (e.g., treewidth and genus) because the graphs of bounded doubling dimension, or diameter can contain the clique of an arbitrary size and thus, are not a subclass of any minor-excluded graphs. Therefore, any result presented in this paper is not a corollary of past results.

For proving our upper bounds, we propose a novel scheme for shortcut construction, known as *short-hop extension*. The simplest form of this scheme is *1-hop extension*, where each node in a part assumes all the incident edges to be the shortcut of its own part. Surprisingly, this very simple construction admits shortcuts for graphs of bounded doubling dimension. For graphs of diameters of three or four, the 2-hop extension (that is, each node in a part takes all the two-length paths starting from itself as the shortcut) clearly yield $O(1)$ dilation, but the second edge in each path suffers from high congestion. Our algorithm circumvents this matter through random choice of the second edges based on hash functions, which is simple though far from triviality to bound the quality of constructed shortcuts. The analytic part includes several new ideas and may be of independent interest.

1.2.2 Exact Maximum Matching Algorithm in the CONGEST Model

In Chapter 3, we consider the *maximum (unweighted) matching* problem in the CONGEST model. For a graph $G = (V(G), E(G))$, finding a set of disjoint edges that do not share any vertices is

[1]The statement of the weighted single-source shortest path problem is slightly simplified. See [50] for details.

Table 1.1: Quality bounds of low-congestion shortcut for specific graph classes.

Graph Family	Quality	Construction Time	Lower bound
General	$\tilde{O}(\sqrt{n}+D)$ [63]	$\tilde{O}(\sqrt{n}+D)$ [63]	$\Omega(\sqrt{n}+D)$ [75]
Planar	$\tilde{O}(D)$ [37]	$\tilde{O}(D)$ [37]	$\tilde{\Omega}(D)$ [37]
Genus-g	$\tilde{O}(\sqrt{g}D)$ [49]	$\tilde{O}(\sqrt{g}D)$ [49]	$\tilde{\Omega}(\sqrt{g}D)$ [49]
Treewidth-k	$\tilde{O}(kD)$ [49]	$\tilde{O}(kD)$ [49]	$\Omega(kD)$ [49]
Clique-width-6	–	–	$\tilde{\Omega}(\sqrt{n}+D)$ (**this paper**)
Expander	$\tilde{O}\left(\tau 2^{O(\sqrt{\log n})}\right)$ [43]*	$\tilde{O}\left(\tau 2^{O(\sqrt{\log n})}\right)$ [43]	–
Doubling Dimension-x	$\tilde{O}(D^x)$ (**this paper**)	$O(1)$ (**this paper**)	–
k-Chordal	$O(kD)$ [57]	$O(1)$ [57]	$\tilde{\Omega}(kD)$ [57]
Minor	$\tilde{O}(D^2)$ [51]	$\tilde{O}(D^2)$ [51]	–
$D=3$	$\tilde{O}(n^{1/4})$ (**this paper**)	$\tilde{O}(n^{1/4})$ (**this paper**)	$\Omega(n^{1/4})$ [67, 76]
$D=4$	$\tilde{O}(n^{1/3})$ (**this paper**)	$\tilde{O}(n^{1/3})$ (**this paper**)	$\Omega(n^{1/3})$ [67, 76]
$5 \leq D \leq \log n$	$\tilde{O}\left(n^{(D-2)/(2D-2)}\right)$ [58]	$\tilde{O}\left(n^{(D-2)/(2D-2)}\right)$ [58]	$\tilde{\Omega}\left(n^{(D-2)/(2D-2)}\right)$ [76]

* τ is the mixing time of network graph G.

called a matching problem, and finding the maximum matching is a fundamental problem in the theory of distributed graph algorithms. Many studies in the context of approximation algorithms provide insight into the globality of the maximum matching problem (see Table 1.2). Lotker et al. [68] presented the first approximation algorithm in the CONGEST model, which is a randomized algorithm to compute $(1-\epsilon)$-approximate maximum matching in $O(\log n)$ rounds for any constant $\epsilon > 0$. The running time of the algorithm depends exponentially on $1/\epsilon$. Bar Yehuda et al. [11] improved the algorithm and proposed an $O(\log \Delta / \log\log \Delta)$-round algorithm of computing $(1-\epsilon)$-approximate matching for any constant $\epsilon > 0$, where Δ is maximum degree of the graph. Kuhn et al. [60] have shown a lower bound of $\Omega(\log \Delta / \log\log \Delta)$ rounds if $\log \Delta \leq \sqrt{\log n}$ holds. Ben-Basat et al. [12] proposed a deterministic $\tilde{O}(s_{\max}^2)$-round CONGEST algorithm. They also proposed a $(1/2-\epsilon)$ approximate algorithm in $\tilde{O}(s_{\max} + (s_{\max}/\epsilon)^2)$ rounds. Ahmadi et al. [6] proposed a deterministic $(2/3-\epsilon)$ approximate maximum matching algorithm in general graphs, which runs in $O(\log \Delta/\epsilon^2 + (\log^2 \Delta + \log^* n)/\epsilon)$ rounds. They also presented an $\tilde{O}(s_{max})$-round algorithm and $O((\log^2 \Delta + \log^* n)/\epsilon)$-round $(1-\epsilon)$ approximate algorithm in bipartite graphs. However, no $o(n^2)$-round algorithm for solving the exact maximum matching problem in the CONGEST model has been proposed so far. In addition, Bacrach et al. [10] pointed out that the bound of $\tilde{\Omega}(\sqrt{n}+D)$ rounds is a strong barrier because the standard framework of two-party communication complexity is unlikely to give any improved lower bound. These observations demonstrate the difficulty of revealing the inherent complexity of the exact maximum matching in the CONGEST model.

The objective of this dissertation is to shed light on the complexity gap of the exact maximum matching problem in the CONGEST model. We present the main theorem of this paper in the CONGEST model below.

Theorem 2. *For any input graph G, there exists a randomized CONGEST algorithm to compute the maximum matching that terminates within $O\left(s_{\max}^{3/2}\right)$ rounds with probability $1 - 1/n^{\Theta(1)}$.*

Our algorithm follows the standard technique of finding *augmenting paths*. If an augmenting path is found, the current matching is improved by flipping the labels of matching edges and non-matching edges along the path. It is well known that the current matching is the maximum if and

Table 1.2: Lower and upper bounds of the maximum matching in the CONGEST model.

Algorithm	Time Complexity	Approximation Level	Remark
Ben-Basat et al. [12]	$\Omega(\lvert s_{\max}\rvert)$	exact	LOCAL
Kuhn et al. [60]	$\Omega\left(\frac{\log\Delta}{\log\log\Delta}\right)$	constant ϵ	$\log\Delta \leq \sqrt{\log n}$
Ben-Basat et al. [12]	$\Omega\left(\frac{1}{\epsilon}\right)$	$1-\epsilon$	LOCAL
Kuhn et al. [61]	$\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$	$1-\epsilon$	LOCAL
Ben-Basat et al. [12]	$\tilde{O}(s_{\max}^2)$	exact	
Ahmadi et al. [6]	$\tilde{O}\left(s_{\max}\right)$	exact	bipartite
Bar-Yehuda et al. [11]	$O\left(\frac{\log\Delta}{\log\log\Delta}\right)$	constant ϵ	
Lotker et al. [68]	$O\left(\frac{2^{2\epsilon^{-2}}\log s_{\max}\log n}{\epsilon^4}\right)$	$1-\epsilon$	
Ahmadi et al. [6]	$O\left(\frac{\log^2\Delta+\log^* n}{\epsilon}\right)$	$1-\epsilon$	bipartite
Ben-Basat et al. [12]	$\tilde{O}\left(s_{\max}+\left(\frac{s_{\max}}{\epsilon}\right)^2\right)$	$\frac{1}{2}-\epsilon$	
Ahmadi et al. [6]	$O\left(\frac{\log\Delta}{\epsilon^2}+\frac{\log^2\Delta+\log^* n}{\epsilon}\right)$	$\frac{2}{3}-\epsilon$	
Our result	$\boldsymbol{O\left(s_{\max}^{3/2}\right)}$	**exact**	

only if there exists no augmenting path in G with respect to the current matching. Hence, the maximum matching problem is reduced to the task of finding augmenting paths $s_{\max}$ times. In the CONGEST model, this approach faces difficulty in the situation where any augmenting path with respect to the current matching is long (i.e., consisting of $\Theta(n)$ edges). It should be emphasized that BFS-like approaches do not work for finding augmenting paths in general graphs because the shortest alternating walk is not necessarily simple because of the existence of odd cycles. The key ingredient of our approach is two new algorithms for finding augmenting paths. They run in $O(\ell^2)$ rounds and $O(s_{\max})$ rounds respectively, where ℓ is the length of the shortest augmenting path for the current matching. Roughly, our algorithm switches between these two algorithms according to the current matching size. The running-time bound is obtained using the following seminal observation by Hopcroft and Karp:

Proposition 1 (Hopcroft and Karp [53])**.** *Given a matching $M \subseteq E$ of a graph G, there always exists an augmenting path of length less than $\lfloor 2s_{\max}/k \rfloor$ if the current matching size is at most the maximum matching size $s_{\max}$ minus k.*

Our augmenting path algorithms utilize Ahmadi and Kuhn's verification algorithm of maximum matching [5], in which each node returns the length of the shortest odd/even alternating paths from a given source (unmatched) node. The construction of the $O(\ell^2)$-round algorithm is relatively straightforward. It is obtained by iteratively finding the predecessor of each node in an augmenting path by sequential $O(\ell)$ invocations of the verification algorithm. The technical highlight of the proposed algorithm is the design of the $O(s_{\max})$-round algorithm. The $O(s_{\max})$-round algorithm constructs a *sparse certificate*, which is a sparse (i.e., containing $O(s_{\max})$ edges) subgraph of G preserving the reachability between two nodes by alternating paths. That is, a sparse certificate contains an augmenting path if and only if the original graph admits an augmenting path. By the sparseness property, a node can collect all the information on the sparse certificate within $O(s_{\max})$ rounds, trivially allowing the centralized solution of finding augmenting paths. To establish a highly parallel construction of sparse certificates, we also propose a new characterization of sparse

certificates, which might also be of independent interest.

1.2.3 Uniform Distribution for Pachinko

Pachinko is a Japanese mechanical gambling game similar to Pinball [1–3]. The machine stands vertically and the player shoots a metal ball into the playfield. Many pins are spiked in the playfield and the ball drops from the top of the field. If the ball goes into a pocket in the field, then the player earns a reward. Recently, Pachinko was analyzed in the context of discrete mathematics. The origin of mathematical Pachinko is based on a book written by Akiyama in 2008 [8]. Recently, Akitaya et al. [7] studied an idealized geometry of a simple form of Pachinko [7]. This study considers one of the mathematical models presented, which is called the *50-50 model*.

The 50-50 model consists of three entities: the field, pins, and a ball. The field is a half-plane triangle grid with the top-side end. A pin can be placed at any grid point. A *row* is a horizontal line where the grid points exist, and a *column* is a vertical line where the grid points exist. Since a triangle grid was considered for this investigation, the intersection points of rows and columns do not necessarily have a grid point (see Figure 1.1). The ball drops from the center of the top-end and falls down vertically. If the ball hits a pin, then it moves to the left or right passage of the pin with an equal probability. Immediately afterwards, the ball continues to fall down vertically. Once the pin arrangement is fixed under the 50-50 model, the probability of dropping the ball in each column can be calculated. In other words, a pin arrangement defines the drop probability distribution for all of the columns. Then, the inverse problem of "deciding if there exists a pin arrangement that generates a given distribution or not" can be considered.

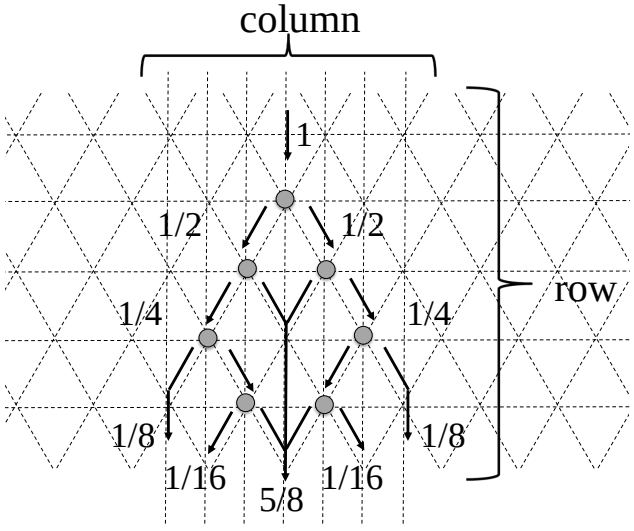

Figure 1.1: An example of the 50-50 model. Each value represents the drop probability of each column.

In [7], it was shown that any probability distribution $\langle p_1, p_2, ..., p_n \rangle$ in the 50-50 model can be constructed within an arbitrarily small additive error; thus, the main theoretical challenge is the generation of the given distribution. The $(1/2^a)$-*uniform distribution* in the 50-50 model is the probability distribution. When the ball drops in the center, the probability is 0 and the probability at the 2^a closest coordinates from the center is $\frac{1}{2^a}$ (see Figure 1.2). Akitaya et al. [7] showed that the $(1/2^a)$-uniform distribution for $a \in \{0, 1, 2, 3, 4\}$ can be constructed. This can also be conjectured

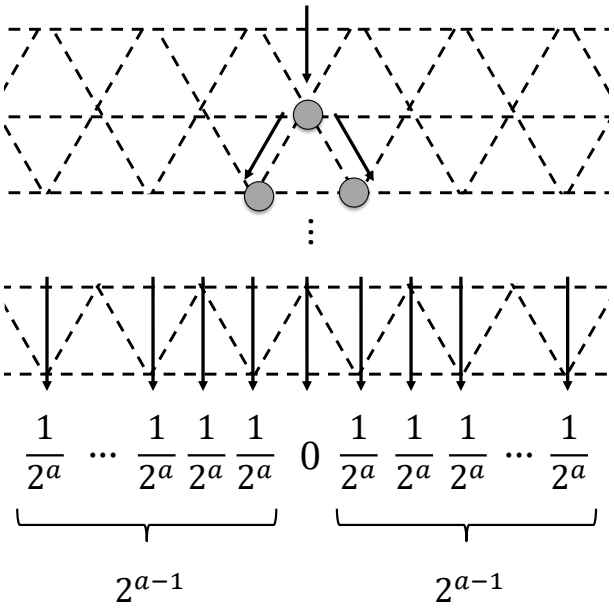

Figure 1.2: An example of the uniform distributions. The center column has a probability of 0.

such that the $(1/2^a)$-uniform distribution for any positive integer a can be constructed. The first contribution of this study is to show that this conjecture is true. In other words, for any $a \geq 1$, this study provides the pin arrangement that generates a $(1/2^a)$-uniform distribution. The number of pins used in the construction is bounded by a polynomial of 2^a. To show the result, a new formulation of the problem is introduced following the notions and terminology of language theory. Even though the language theory is simple, it is substantially useful for the analysis of the 50-50 model.

As the second contribution, a computational-complexity aspect of the 50-50 model was also considered. Since the pin arrangement in the 50-50 model corresponds to a transformation from a given probability distribution (for all x-coordinates (i.e. $\mathbb{Z}^2$)) to another one, the matter of its design naturally yields one decision problem. For any two input distributions A and B, is there a pin arrangement that transforms A to B? This study focuses on the computational complexity of this decision problem. Unfortunately, this study does not determine any hardness results for this problem. Instead, for a slight variant of it, where B can be *partial* in the sense that B specifies the drop probability only for a subset of all columns, this study uses non-deterministic polynomial-time (NP) hardness to decide the transformability from A to B.

1.3 Related Work

1.3.1 Low-Congestion Shortcut and Graph Parameters

The MST problem is one of the most fundamental problems in distributed graph algorithms. As well as its own importance, MST has several applications for solving other graph problems in distributed settings (e.g., detecting connected components and minimum cut). Several studies have addressed the design of efficient MST algorithms in the CONGEST model [34, 35, 40, 45, 47, 55, 63, 73, 74], and the round-complexity of MST construction is a central topic in distributed

[2]The notation $\mathbb{Z}$ is the field of the integer numbers.

complexity theory [27, 28, 67, 72, 75, 76]. The inherent difficulty of MST construction is solving the partwise aggregation (minimum) problem efficiently. This viewpoint was first identified by Ghaffari and Haeupler [37] explicitly, as well as an efficient algorithm for solving MST construction in planar graphs. The concept of low-congestion shortcut is newly invented herein for encapsulating the difficulty of partwise aggregation. Recently, several follow-up papers have been published to extend the applicability of low-congestion shortcut, which break the known general lower bounds of MST and its applications in specific graph classes: This line covers bounded-genus graphs [37, 49], bounded-treewidth graphs [49], graphs with excluded minors [51], and expander graphs [42, 43] (see Table 1.1). All the shortcuts stated here belong to the class of *tree-restricted* shortcuts, where the shortcut edges augmented to each part are a subgraph of a precomputed spanning tree (typically a breadth first search (BFS) tree). It is shown that there exists a universal algorithm for computing tree-restricted shortcuts [48]. To the best of our knowledge, the upper bounds presented in this paper are the first to exhibit non-trivial shortcuts not belonging to the tree-restricted class. The application of low-congestion shortcut is not limited to MST. As stated in Theorem 1, low-congestion shortcut also admits efficient solutions for an approximate minimum cut, and a single-source shortest path. In addition, a few algorithms utilize low-congestion shortcut as an important building block, e.g., the depth first search in planar graphs [50], approximate tree decomposition [65], along with diameter and distance labeling scheme in planar graphs [66]. Haeupler et al. [47] show a message-reduction scheme of shortcut-based algorithms, which drops the total number of messages exchanged by the algorithm into $\tilde{O}(m)$, where m denotes the number of links in the network. On the negative side, it is known that the hardness of (approximate) a diameter cannot be encapsulated by low-congestion shortcut. Abboud et al. [4] showed a hard-core family of unweighted graphs with $O(\log n)$ treewidth, where any diameter computation in the CONGEST model requires $\tilde{\Omega}(n)$ rounds. Because any graph with $O(\log n)$ treewidth admits a low-congestion shortcut of quality $\tilde{O}(D)$, this result implies that it is not possible to compute the diameter of graphs efficiently by using only the property of low-congestion shortcuts.

Although our results exhibit a tight upper bound for graphs of diameter three or four, a more generalized lower bound is known for small-diameter graphs. [76]. For any $\log n \geq D \geq 3$, it is proved that there exists a network topology that incurs the $\tilde{\Omega}\left(n^{(D-2)/(2D-2)}\right)$-round time complexity for any MST algorithm. In more restricted cases of $D = 1$ and $D = 2$, Jurdzinski et al. [55] and Lotker et al. [67] showed $O(1)$- and $O(\log n)$-round MST algorithms, respectively. Recently, Kogan et. al. shows the upper bound of the MST in constant diameter graphs [58].

1.3.2 Maximum Matching Algorithm

The maximum matching has been studied for both the distributed algorithm and the centralized algorithm. The LOCAL model is a model for distributed graph algorithm and is like the CONGEST model with arbitrarily large messages. In the LOCAL model, it is known that no $o(1/\epsilon)$ algorithm exists for the $(1-\epsilon)$-approximate maximum matching problem [12]. Together with the $\Omega(\sqrt{\log n/\log\log n})$-round lower bound reported by Kuhn et al. [61], the lower bound in the LOCAL model is obtained as $\Omega(1/\epsilon + \sqrt{\log n/\log\log n}) = ((\log n)/\epsilon)^{\Omega(1)}$. Ghaffari et al. [41] showed a $((\log n)/\epsilon)^{O(1)}$ upper bound for the $(1-\epsilon)$ approximate maximum matching problem. By combining these results, we infer that the time complexity of solving the $(1-\epsilon)$ approximate maximum matching problem is $(\log n/\epsilon)^{\Theta(1)}$ in the LOCAL model. Ben-Basat et al. also proved the lower bound of the maximum matching as $\Omega(s_{max})$ in the LOCAL model [12].

In addition to distributed computing, many studies have considered centralized exact maximum matching algorithms. Edmonds presented the first centralized polynomial-time algorithm for the maximum matching problem [25, 26] by following the seminal blossom argument. Hopcroft and

Karp proposed a phase-based algorithm of finding multiple augmenting paths [53]. Their algorithm finds a maximal set of pairwise disjoint shortest augmenting paths in each phase. They showed that $O(\sqrt{n})$ phases suffice to compute the maximum matching and proposed an algorithm implementing one phase in $O(m)$ time for bipartite graphs. Several studies have reported phase-based algorithms for general graphs that attain $O(\sqrt{n}m)$ time [14, 33, 78].

1.3.3 Uniform Distribution for Pachinko

Many of mathematical puzzles such as *15-puzzles* and *sokoban* can be seen as a *transformation problem* of a local interaction systems [22, 46]. In addition, some puzzles can be seen as a *reconfiguration problem* which is the kind of the transformation problem. The reconfiguration problem has been studied not only in the field of the mathematical puzzle but also in the field of the graph theory [77]. The reconfiguration problems are concerned with relationships among two solutions of a problem instance, where the reconfiguration of one solution to another is a sequence of steps such that each step produces an intermediate feasible solution. The reconfiguration problems has been studied for some well-known problems that includes independent set [15,23,31,56,80], set cover [69], minimum spanning tree, matching [54] and so on. Unfortunately, many decision problems related to reconfiguration problems are known to be NP-hard or PSPACE-complete. On the other hand, some polynomial time algorithms are shown in the restricted case [15, 23, 31, 56].

The origin of mathematical Pachinko is based on a book written by Akiyama in 2008 [8]. Recently, Akitaya et al. [7] studied an idealized geometry of a simple form of Pachinko [7]. In [7], it was shown that any probability distribution $\langle p_1, p_2, ..., p_n \rangle$ in the 50-50 model can be constructed within an arbitrarily small additive error. Akitaya et al. [7] showed that the $(1/2^a)$-uniform distribution for $a \in \{0, 1, 2, 3, 4\}$ can be constructed. This can also be conjectured such that the $(1/2^a)$-uniform distribution for any positive integer a can be constructed.

1.4 Structure of This Dissertation

The dissertation is organized as follows. In Chapter 2, we present a low-congestion shortcut construction for graphs of bounding doubling dimensions and graphs of diameters of three or four in the CONGEST model. Moreover, we show the lower bounds of low congestion shortcut in constant clique-width graphs in the CONGEST model. In Chapter 3, we present a maximum matching in the CONGEST model. In Chapter 4, we provide an explicit construction of the $(1/2^a)$-uniform distribution for any $a \geq 1$ and its analysis for the number of pins. Moreover, we present the NP-hardness results for the transformability of the distributions

Chapter 2

Low-Congestion Shortcut and Graph Parameters

2.1 Introduction

In this chapter, we consider the relationship between several major graph parameters and the quality of low-congestion shortcut in the CONGEST model. In particular, we focus on the following three parameters, that is: (1) doubling dimension, (2) diameter, and (3) clique width. The precise statement of our results is as follows:

- There is an $O(1)$-round algorithm that constructs a low congestion shortcut with quality $\tilde{O}(D^x)$ for any doubling dimension-x graph.
- There exists an algorithm for constructing a low-congestion shortcut with quality $\tilde{O}(n^{1/4})$ in $\tilde{O}(n^{1/4})$ rounds for any graph of diameter three. In addition, there exists an algorithm for constructing a low-congestion shortcut with quality $\tilde{O}(n^{1/3})$ in $\tilde{O}(n^{1/3})$ rounds for any graph of diameter four.
- We present a negative instance certifying that bounded clique-width does not help in the construction of good-quality shortcuts. More precisely, we provide an instance of clique-width six, where the construction of MST is as expensive as the general case, i.e., $\tilde{\Omega}(\sqrt{n}+D)$ rounds.

2.2 Preliminaries

2.2.1 CONGEST model

In this section, we described a general definition of the *CONGEST* model. The vertex set and edge set of a given graph G are, respectively, denoted by $V(G)$ and $E(G)$. A distributed system is represented by a simple undirected connected graph $G = (V(G), E(G))$. Let n and m be the numbers of nodes and edges, respectively. The diameter of a given subgraph $H \subseteq G$ is denoted by $D(H)$. Nodes and edges are uniquely identified by integer values, which are represented by $O(\log n)$ bits. The set of edges incident to $v \in V(G)$ is denoted by $I_G(v)$. In the CONGEST model, the computation is done in synchronous rounds. In one round, each node v sends and receives $O(\log n)$-bit messages through the edges in $I_G(v)$ and executes local computation following its internal state, local random bits, and received messages. It is guaranteed that every message

sent in a round is delivered to the destination within the same round. Each node has no prior knowledge of the network topology, except for its neighborhood IDs. We use the labeling of nodes and/or edges for specifying inputs and outputs of algorithms. Each node has information on the label(s) assigned to itself and those assigned to its incident edges. Let $N(v)$ be the set of nodes that are adjacent to v in G, and let $N^+(v) = N(v) \cup \{v\}$. We define $N(S) = \cup_{s \in S} N(s)$, and $N^+(S) = \cup_{s \in S} N^+(s)$, for any $S \subseteq V$. For two node subsets $X, Y \subseteq V$, we also define $E(X, Y) = \{(u, v) \in E \mid u \in X, v \in Y\}$. If X (resp. Y) is a singleton $X = \{w\}$, (resp. $Y = \{w\}$), we describe $E(X, Y)$ as $E(w, Y)$ (resp. $E(X, w)$). The *distance* (that is, the number of edges in the shortest path) between two nodes u and v in G is denoted by $\mathrm{dist}_G(u, v)$. A *walk* W of G is an alternating sequence $W = \{v_0, e_1, v_1, e_2, \ldots, e_\ell, v_\ell\}$ of vertices and edges such that $e_i = (v_{i-1}, v_i)$, $v_i \in V(G)$, and $e_i \in E(G)$ holds for any $1 \leq i \leq \ell$. The *length* of the walk W is a number of edges in W. With a small abuse of notations, we treat a walk W as the sequence of nodes or edges representing the path, as the set of nodes or edges in the path, or the subgraph of G forming the path. A walk $W = \{v_0, v_1, \ldots, v_\ell\}$ is called a (simple) *path* if every vertex in W is distinct. A path $U = \{v_0, v_1 \ldots, v_\ell\}$ is referred to as *chordless* if and only if for any two nodes $v_i, v_j \in U$ and $|i - j| \geq 2$, it holds that $(v_i, v_j) \notin E(G)$ For any walk $W = \{v_0, v_1, \ldots, v_\ell\}$ of G, we define $W \circ u$ as the walk obtained by adding u, satisfying $(v_\ell, u) \in E(G)$, to the tail of W. For any edge $e = (v_\ell, u)$, we also define $W \circ e = W \circ u$. Given a walk W containing a node u, we denote by W^p_u and W^s_u the prefix of W up to u and the suffix of W from u, respectively. We also denote the inversion of the walk $W = \{v_0, v_1, \ldots, v_\ell\}$ (i.e., the walk $\{v_\ell, v_{\ell-1}, \ldots, v_0\}$) by $\overline{W}$. The length of a walk W is represented by $|W|$.

2.2.2 Partwise Aggregation

The *partwise aggregation* is a communication abstraction defined over a set $\mathcal{P} = \{P_1, P_2, \ldots, P_N\}$ of mutually disjoint and connected subgraphs known as *parts*, and provides simultaneous fast group communication among the nodes in each P_i. It is formally defined as follows:

Definition 1 (Partwise Aggregation (PA))**.** *Let $\mathcal{P} = \{P_1, P_2, \ldots, P_N\}$ be the set of connected mutually-disjoint subgraphs of G, and each node $v \in V(P_i)$ maintains variable b^i_v storing an input value $x^i_v \in X$. The output of the partwise aggregation problem is to assign $\oplus_{w \in P_i} x^i_w$ with b^i_v for any $v \in V(P_i)$, where $\oplus$ is an arbitrary associative and commutative binary operation over X.*

The straightforward solution of the partwise aggregation problem in the CONGEST model is to perform the convergecast and broadcast in each part P_i independently. In particular, we construct a BFS tree for each part P_i (after the selection of the root by any leader election algorithm). The time complexity is proportional to the diameter of each part P_i, which can be large ($\Omega(n)$ in the worst case) independently of the diameter of G.

2.2.3 (d, c)-Shortcut

As we stated in the introduction, the notion of low-congestion shortcut is introduced for quickly solving the partwise aggregation problem (for some specific graph classes). Formal definition of (d, c)-shortcuts is provided as follows:

Definition 2. *[Ghaffari and Haeupler [37]] Given a graph $G = (V(G), E(G))$ and partition $\mathcal{P} = \{P_1, P_2, \ldots, P_N\}$, of G into node-disjoint and connected subgraphs, we define a (d, c)-shortcut of G and $\mathcal{P}$ as a set of subgraphs $\mathcal{H} = \{H_1, H_2, \ldots, H_N\}$ of G such that;*

1. *For each i, the diameter of $P_i + H_i$ is at most d (d-dilation).*

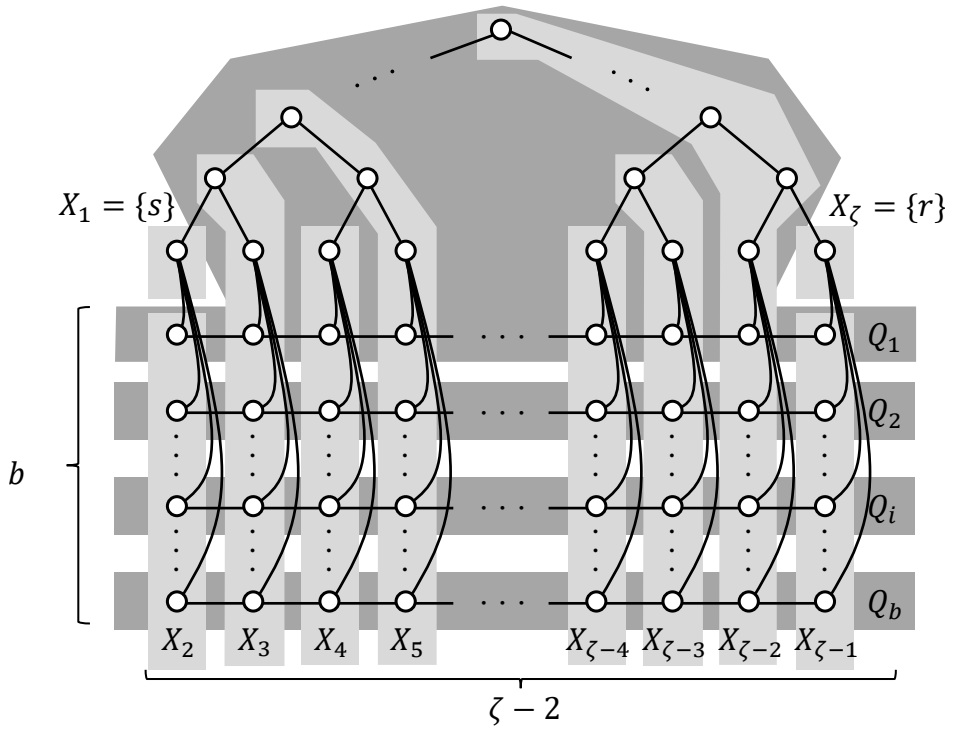

Figure 2.1: Example of $\mathcal{G}(O(\zeta b), b, \zeta, O(\log n))$.

2. *For each edge $e \in E(G)$, the number of subgraphs $P_i + H_i$ containing e is at most c (c-congestion).*

The values of d and c for a (d, c)-shortcut $\mathcal{H}$ are known as the *dilation* and *congestion* of $\mathcal{H}$. As a general statement, a (d, c)-shortcut that is constructed in f rounds admits the solution of the partwise aggregation problem in $\tilde{O}(d + c + f)$ rounds [36, 37]. Because the parameter $d + c$ asymptotically affects the performance of the application, we refer to the value of $d+c$ as the *quality* of (d, c)-shortcuts. A low-congestion shortcut with quality q is simply known as a *q-shortcut*.

2.2.4 Lower-Bound Framework

To prove the lower bound of MST, we introduce a simplified version of the framework by Das Sarma et al. [76]. In this framework, we consider the graph class $\mathcal{G}(n, b, \zeta, c)$ that is defined below. A vertex set $X \subseteq V(G)$ is known as *connected* if the subgraph induced by X is connected.

Definition 3. *For $n, b, c \geq 0$ and $\zeta \geq 3$, the graph class $\mathcal{G}(n, b, \zeta, c)$ is defined as the set of n-vertex graphs $G = (V(G), E(G))$ satisfying the following conditions:*

- ***(C1)** The vertex set $V(G)$ is partitioned into ζ disjoint vertex sets $\mathcal{X} = \{X_1, X_2, \ldots, X_\zeta\}$, such that X_1 and X_ζ, are singletons (let $X_1 = \{s\}$ and $X_\zeta = \{r\}$).*
- ***(C2)** The vertex set $V(G)\backslash\{s, r\}$ is partitioned into b disjoint connected sets $\mathcal{Q} = \{Q_1, \ldots, Q_b\}$ such that $|E(X_1, Q_i)| \geq 1$ and $|E(X_l, Q_i)| \geq 1$ hold for any $1 \leq i \leq b$.*
- ***(C3)** Let $R_i = \bigcup_{i+1 \leq j \leq \zeta} X_j$ and $L_i = \bigcup_{0 \leq j \leq \zeta - i} X_j$. For $2 \leq i \leq \zeta/2 - 1$, $|E(R_i, N(R_i) \setminus R_{i-1})| \leq c$, and $|E(L_i, N(L_i) \setminus L_{i-1})| \leq c$.*

Figure 2.1 shows the vertex partition $\mathcal{X}$ and $\mathcal{Q}$ for the hard-core instances presented in the original proof by Das Sarma et al. [76]. This graph belongs to $\mathcal{G}(O(\zeta b), b, \zeta, O(\log n))$. For class $\mathcal{G}(n, b, \zeta, c)$, the following theorem holds, which is just a corollary of the result by Das Sarma et al. [76]:

Theorem 3 (Das Sarma et al. [76])**.** *For any graph $G \in \mathcal{G}(n, b, \zeta, c)$ and any MST algorithm A, there exists an edge-weight function $w_{A,G} : E \to \mathbb{N}$ such that the execution of A in G requires $\tilde{\Omega}(\min\{b/c, \zeta/2 - 1\})$ rounds. This bound holds with high probability even if A is a randomized algorithm.*

2.2.5 1-Hop Extension Scheme

Throughout this study we utilize the *1-hop extension* scheme for shortcut construction, which is stated as follows:

> For any $V_{P_i} \subseteq V(G)$, node $v \in V_{P_i}$ adds each incident edge (v, u) to H_i and informs u of $(v, u) \in H_i$.

It is trivial to implement this scheme using only one round in the CONGEST model. Because each node belongs to one part, the congestion of each edge is at most two. Hence, the technical challenge of this scheme is bound dilation. For the proof, we introduce the concept of (a, b)*-path dominating set*, which characterizes the graphs allowing good shortcuts through 1-hop extension.

Definition 4. *Given a path $U \subseteq G$, a (a, b)-path dominating set $S \subseteq V_G$ of U is a node subset satisfying the following two conditions:*

- *For any $u \in V_U$, there exists $s \in N^+(S)$ such that $\mathrm{dist}_U(u, s) \leq a$ holds.*
- $|S| \leq b$.

It is easy to check that if S is a (a, b)-path dominating set of U, $S \cap N^+(U)$ is also a (a, b)-path dominating set of U. Thus, in the following argument, we assume that any (a, b)-path dominating set for U is a subset of $N^+(U)$ without loss of generality. We say that G is (a, b)*-path dominating* if and only if any chordless path $U \subseteq G$ has a (a, b)-path dominating set. By definition, any graph having a dominating set of size b is $(0, b)$-path dominating.

Lemma 1. *The 1-hop extension constructs an $O((a+1)b)$-shortcut for any (a, b)-path dominating graph.*

Proof. Because the congestion bound is trivial, we focus on bounding dilation. Let G be any (a, b)-path dominating graph, P_i be any part of G, and H_i is the shortcut through 1-hop extension for part P_i. Let $U = (s_0, s_1, \ldots, s_\ell)$ be any shortest path in P_i. Because U is the shortest, it is chordless, and thus it has a (a, b)-path dominating set S_U of size $b' \leq b$. Let $Z_U = (V_{Z_U}, E_{Z_U})$ be the subgraph of G such that $V_{Z_U} = U \cup S_U$ and $E_{Z_U} = E(U, U) \cup E(U, S_U)$ holds. Because $S_U \subseteq N^+(U) \subseteq N^+(P_i)$, every edge in $E(U, S_U)$ is a shortcut for H_i. Thus, to prove the lemma, it suffices to show that $\mathrm{dist}_{Z_U}(s_0, s_\ell) = O((a+1)b')$, for any U. The proof is by the induction on b', that is, we show that every chordless path in P_i having (a, b')-path dominating set S_U of size $b' \leq b$ satisfying $\mathrm{dist}_{Z_U}(s_i, s_j)$ is at most $(2a+3)b'$ for all $b' \leq b$. (Basis) The case of $b' = 1$: Let w be the unique node in S_U, i be the minimum index such that $s_i \in N^+(w)$ holds, and j be the maximum index such that $s_j \in N^+(w)$ holds. Because $S_U = \{w\}$ is a $(a, 1)$-path dominating set, we obtain $\mathrm{dist}_{Z_U}(s_0, s_\ell) \leq \mathrm{dist}_{Z_U}(s_0, s_i) + \mathrm{dist}_{Z_U}(s_j, s_\ell) + 2 \leq 2a + 2$. (Inductive Step) Suppose

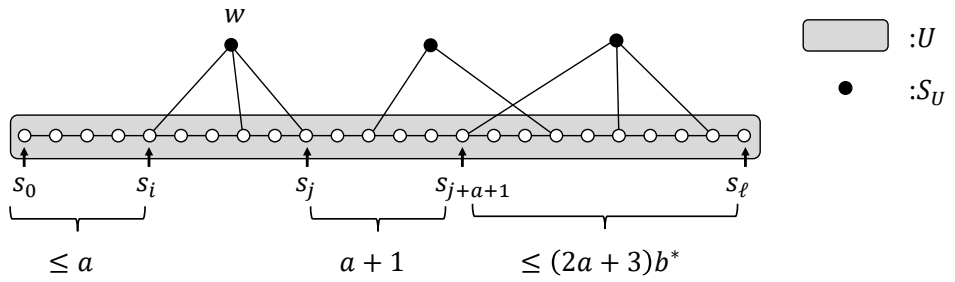

Figure 2.2: Proof of Lemma 1.

as the induction hypothesis that any chordless path $U' = (s'_0, s'_1, \dots s'_{\ell'})$ in P_i having (a, b'')-path dominating set of size $b'' < b'$ satisfies $\mathrm{dist}_{Z_{U'}}(s'_0, s'_{\ell'}) \leq (2a+3)b''$. Let i be the minimum index such that $s_i \in N^+(S_U)$ holds, w is any node in $S_U \cap N^+(s_i)$, and j be the maximum index such that $s_j \in N^+(w)$ (see Fig. 2.2). If $\ell - j \leq a$, we obtain $\mathrm{dist}_{Z_U}(s_0, s_\ell) \leq \mathrm{dist}_{Z_U}(s_0, s_i) + \mathrm{dist}_{Z_U}(s_j, s_\ell) + 2 \leq 2a+2$. In the case of $\ell - j > a$, any node s_h for $j+a+1 \leq h \leq \ell$ has no node $s_{h'}$ such that $s_{h'} \in N^+(w)$ and $|h'-h| \leq a$ hold. Hence, the vertex set $S_U \backslash \{w\}$ is a (a, b^*)-path dominating set of (chordless) subpath $U^* = (s_{j+a+1}, \dots, s_\ell)$. Because $b^* < b'$ holds, by the induction hypothesis, we obtain $\mathrm{dist}_{Z_U}(s_{j+a+1}, s_\ell) \leq \mathrm{dist}_{Z_{U^*}}(s_{j+a+1}, s_\ell) \leq (2a+3)b^*$. It follows that $\mathrm{dist}_{Z_U}(s_0, s_\ell) \leq \mathrm{dist}_{Z_U}(u, s_i) + \mathrm{dist}_{Z_U}(s_j, s_{j+a+1}) + \mathrm{dist}_{Z_U}(s_{j+a+1}, s_\ell) + 2 \leq a + (a+1) + (2a+3)b^* + 2 \leq (2a+3)b'$. The lemma holds. □

2.3 Low-Congestion Shortcut for Constant Doubling Dimension Graphs

A pair of a set $V(G)$ and the associated function $\mathrm{dist} : V(G) \times V(G) \to \mathbb{R}$ is known as a metric space if and only if the following three conditions hold: (1) $\mathrm{dist}(u, v) = 0$ if and only if $v = u$, (2) $\mathrm{dist}(u, v) = \mathrm{dist}(v, u)$ for all $u, v \in V$, and (3) $\mathrm{dist}(u, v) \leq \mathrm{dist}(u, w) + \mathrm{dist}(w, v)$ for all $u, v, w \in V(G)$. The doubling dimension of a metric space V is the smallest positive integer x such that it is possible to cover the ball $B(v, r) = \{u \mid \mathrm{dist}(v, u) < r\}$ of radius r with the union of at most 2^x balls of radius $r/2$ for any $v \in V(G)$ and $r > 0$. A graph $G = (V(G), E(G))$ has a doubling dimension x if $(V(G), \mathrm{dist}_G)$ is a metric space of the doubling dimension x. The graphs of bounded doubling dimensions can be assumed to be a generalization of unit disk graphs, and are often considered in the context of distributed computing [21, 29, 59, 62]. The main results of this section are that the graphs of bounded doubling dimensions allow a good shortcut. We have the following theorem:

Theorem 4. *Let x be the doubling dimension of the graph G. Then, there is an $O(1)$-round algorithm that constructs low-congestion shortcut with quality $\tilde{O}(D^x)$.*

The theorem is obtained by combining the following lemma with Lemma 1. Recall that any graph having a dominating set of size $\tilde{O}(D^x)$ is $(0, \tilde{O}(D^x))$-path dominating.

Lemma 2. *Let G be any graph of the doubling dimension x. There is a dominating set of size $\tilde{O}(D^x)$ in graph G.*

Proof. We show that G is covered by at most 2^{ix} balls with radius $(D/2^i)$ for any $1 \leq i \leq \log D$. The lemma is obtained by setting $i = \log D$. The proof follows the induction on i. (Basis) The case of $i = 1$ is obtained from the definition of the doubling dimension. (Inductive step) Suppose as the induction hypothesis that there exists at most 2^{ix} balls with radius $D/2^i$ that cover the graph G.

By the definition of the doubling dimension, each ball with radius $D/2^i$ can be covered at most by 2^x balls with radius $D/2^{i+1}$. Therefore, there exist at most $2^{(i+1)x}$ balls with radius $D/2^{i+1}$ that cover the graph G. The lemma is proved. □

2.4 Low-Congestion Shortcut for Small Diameter Graphs

Let $\kappa_D = n^{(D-2)/(2D-2)}$. Note that $\kappa_3 = n^{1/4}$, and $\kappa_4 = n^{1/3}$ hold. The main result in this section is the following theorem:

Theorem 5. *For any graph of diameter $D \in \{3, 4\}$, there exists an algorithm for constructing low-congestion shortcut with quality $\tilde{O}(\kappa_D)$ in $\tilde{O}(\kappa_D)$ rounds.*

2.4.1 Centralized Construction

In the following argument, we use terminology "whp." (with high probability) to mean that the event considered occurs with probability $1 - n^{-\omega(1)}$ (or equivalently $1 - e^{-\omega(\log n)}$). For simplicity of the proof, we treat any whp. event as if it necessarily occurs. (i.e., with P=1). Because the analysis below handles only a polynomially bounded number of whp. events, the standard union-bound argument guarantees that everything simultaneously occurs whp; that is, any consequence yielded by the analysis also occurs whp. Because the proof is constructive, we first present the algorithms for $D = 3$ and 4. They are described as a (unified) centralized algorithm, and the distributed implementation is explained later. Let N' be the number of parts whose diameter is greater than $12\kappa_D \log^3 n$ (say *large* part). Assume that $P_1, P_2, \ldots, P_{N'}$ are large without loss of generality. Because each part P_i ($1 \leq i \leq N'$) contains at least κ_D nodes, $N' \leq n/\kappa_D$ holds clearly. Our technical challenge is to reduce the dilation of the large part. To this end, we separate the large part into the subparts whose diameters are $\tilde{O}(\kappa_D)$, and shows that the shortcut edges establish at least one length-D path between any two subparts. Note that this separation scheme is introduced only for the analysis, and the algorithm does not actually construct it. The detailed explanation of the scheme is explained later. First, each large part computes 1-hop extension. As shown in the previous section, the 1-hop extension only increases the congestion by $O(1)$. Therefore, it suffices to show that at least one shortcut path of length $D - 2$ is established between any two extended subparts. For the case of $D = 3$, the independent sampling of each edge with probability $1/n^{1/2}$ guarantees the construction of such paths (of length $D-2 = 1$). For the case of $D = 4$, we introduce a new edge sampling scheme based on hash functions of limited independence, which positively correlates two edges incident to a common vertex, and thus amplifies the probability of establishing length-2 shortcut paths without too much increase of congestion. The precise description of the algorithms is stated below. It is applied to each large part P_i for the construction of H_i.

1. Each node $v \in V_{P_i}$ adds its incident edges to H_i (i.e., compute the 1-hop extension).
2. This step adopts two different strategies according to the value of D. ($D = 3$) Each node $u \in N^+(V_{P_i})$ adds each incident edge (u, v) to H_i with probability $1/n^{1/2}$. ($D = 4$). Let $\mathcal{Y} = [1, n^{1/3}/\log n]$. We first prepare an $(n^{1/3} \log^3 n)$-wise independent hash function $h : [0, N-1] \times V \to \mathcal{Y}$, [1]. At node $u \in V$, each incident edge (u, v) satisfying $v \in N^+(V_{P_i})$ is independently sampled with probability $1/h(u, i)$. All the sampled edges are added to H_i.

[1] Let X and Y, be two finite sets. For any integer $k \geq 1$, a family of hash functions $\mathcal{H} = \{h_1, h_2, \ldots, h_p\}$, where each h_i is a function from X to Y, is known as *k-wise independent* if for any distinct $x_1, x_2, \ldots, x_k \in X$ and any $y_1, y_2, \ldots y_k \in Y$, a function h sampled from $\mathcal{H}$ uniformly at random satisfies $\Pr[\bigwedge_{1 \leq i \leq k} h(x_i) = y_i] = 1/|Y|^k$.

We show that this algorithm provides a low-congestion shortcut of quality $\tilde{O}(\kappa_D)$. First, we observe the bound for congestion. Let H_i^1 be the set of the edges added to H_i in the first step, and H_i^2 be those added in the second step. Because the congestion of the 1-hop extension is negligibly small, it suffices to consider the congestion incurred by step 2. Intuitively, we can believe the congestion of $\tilde{O}(\kappa_D)$ as the expected congestion of each edge is $\tilde{O}(\kappa_D)$: Because the total number of large parts is at most n/κ_D, the expected congestion of each edge incurred in step 2 is $n/\kappa_D \cdot (1/n^{1/2}) = O(n^{1/4})$ for $D = 3$, and $(n/\kappa_D)\sum_{y\in\mathcal{Y}}(1/y)\cdot(1/|\mathcal{Y}|) \leq (n/\kappa_D)\cdot(\log n/|\mathcal{Y}|) = \tilde{O}(n^{1/3})$ for $D = 4$.

Lemma 3. *The congestion of the constructed shortcut is $\tilde{O}(\kappa_D)$ whp.*

Proof. It suffices to show that the congestion of any edge $e = (u, v) \in E(G)$ is $\tilde{O}(\kappa_D)$, whp. For simplicity of proof, we observe an undirected edge $e = (u, v)$ as two (directed) edges (u, v) and (v, u), and distinguish the events by adding (u, v) to shortcuts by u and that by v; that is, the former is recognized as adding (u, v), whereas the latter is recognized by adding (v, u). Clearly, the asymptotic bound holding for directed edge (u, v) also holds for the corresponding undirected edge (u, v) actually existing in G (which is at most twice of the directed bound). Because the first step of the algorithm increases the congestion of each directed edge at most by one, it suffices to show that the congestion incurred by the second step is at most $\tilde{O}(\kappa_D)$.

Let X_i be the indicator random variable for event $(u, v) \in H_i^2$, and $X = \sum_i X_i$. The goal of the proof is to show that $X = \tilde{O}(\kappa_D)$ holds whp. The cases of $D = 3$ and $D = 4$ are proved separately. ($D = 3$) Because at most n/κ_3 large parts exist, we have that $\mathbb{E}[X] \leq (n/\kappa_3)\cdot(1/n^{1/2}) = n^{1/4} = \kappa_3$. The straightforward application of the Chernoff bound to X allows us to bound the congestion of e by at most $2\kappa_3$ with probability $1 - e^{-\Omega(n^{1/4})}$. ($D = 4$) Let $\mathcal{P}'$ be the subset of all large parts P_j such that $u \in N^+(P_j)$ holds. Consider an arbitrary partition of $\mathcal{P}'$ into several groups with a size of at least $(n^{1/3}\log^3 n)/2$ and at most $n^{1/3}\log^3 n$. Let q be the number of groups. Each group was identified by a number $\ell \in [1, q]$. We refer to the ℓ-th group as $\mathcal{P}^\ell$. Fixing ℓ, we bound the number of parts in $\mathcal{P}^\ell$ using $e = (u, v)$ as the shortcut edge. Let Y_i be the value of $h(u, i)$. For $P_i \in \mathcal{P}^\ell$, the probability that $X_i = 1$ is

$$\begin{aligned}\Pr[X_i = 1] &= \sum_{y\in\mathcal{Y}} \Pr[Y_i = y]\frac{1}{y} \\ &= \frac{Har\,(|\mathcal{Y}|)}{|\mathcal{Y}|},\end{aligned}$$

where $Har(x)$ is the harmonic number of x, i.e., $\sum_{1\leq i\leq x} i^{-1}$. Letting $X^\ell = \sum_{j\in P^\ell} X_j$, we have $\mathbb{E}[X^\ell] = (|P^\ell|Har(|\mathcal{Y}|))/|\mathcal{Y}|$. Because $Har(x) \geq 1$, we have $\mathbb{E}[X^\ell] \geq |P^\ell|/|\mathcal{Y}| = (\log^4 n)/2$. As the hash function h is $(n^{1/3}\log^3 n)$-wise independent, it is easy to check that $X_1, X_2, \ldots, X_{p^\ell}$ are independent. We apply Chernoff bound to X^ℓ and obtain $\Pr[X^\ell \leq 2\mathbb{E}[X^\ell]] \geq 1 - e^{-\Omega(\mathbb{E}[X^\ell])} = 1 - e^{-\Omega(\log^4 n)}$. It implies that for any ℓ, at most $2\mathbb{E}[X^\ell]$ groups use (u, v) as their shortcut edges. The total congestion of (u, v) is obtained by summing up $2\mathbb{E}[X^\ell]$ for all $\ell \in [1, q]$, which results in the following:

$$\begin{aligned}\sum_\ell 2\mathbb{E}[X^\ell] &\leq \sum_\ell \frac{2|P^\ell|\log n}{|\mathcal{Y}|} \\ &= \frac{2|\mathcal{P}'|\log n}{|\mathcal{Y}|} \\ &= \tilde{O}(n^{1/3}).\end{aligned}$$

The lemma is proved. □

For bounding dilation, we first introduce several preliminary notions and terminologies. Given a graph $G = (V(G), E(G))$, a subset $S \subset V(G)$ is known as an (α, β)-*ruling set* if it satisfies the following: (1) for any $u, v \in S$, $\mathrm{dist}_G(u, v) \geq \alpha$ holds, and (2) for any node $v \in V(G)$, there exists $u \in S$ such that $\mathrm{dist}_G(v, u) \leq \beta$ holds. It is known that there exists an $(\alpha, \alpha+1)$-ruling set for any graph G [9]. Let $\hat{P}_i = P_i + H_i^1$ for short. For the analysis of P_i's dilation, it suffices to consider the case where the diameter of $\hat{P}_i$ is greater than $12\kappa_D \log^3 n$. We first consider an $(\alpha, \alpha+1)$-ruling set of $\hat{P}_i$ for $\alpha = 12\kappa_D \log^3 n$, which is denoted by $S = \{s_0, s_1, \ldots, s_z\}$. Note that this ruling set is introduced only for the analysis, and the algorithm does not actually construct it. The key observation of the proof is that for any s_j $(1 \leq j \leq z)$ H_i contains a path of length $\tilde{O}(\kappa_D)$ from s_0 to s_j whp. It follows that any two nodes $u, v \in V_{\hat{P}_i}$ are connected by a path of length $\tilde{O}(\kappa_D)$ in $P_i + H_i$ because any node in $V_{\hat{P}_i}$ has at least one ruling set node within distance $\alpha + 1$ in $P_i + H_i^1$.

To prove the above-mentioned claim, we further introduce the notion of *terminal sets*. A terminal set $T_j \subseteq V_{P_i}$ associated with $s_j \in S$ $(0 \leq j \leq z)$ is the subset of V_{P_i} satisfying (1) $|T_j| \geq \kappa_D \log^3 n$, (2) $\mathrm{dist}_{P_i+H_i}(s_j, x) \leq 6\kappa_D \log^3 n$ for any $x \in T_j$, and (3) $N^+(x) \cap N^+(y) = \emptyset$ for any $x, y \in T_j$ (note that $N^+(\cdot)$ is the set of neighbors in G, not in $P_i + H_i^1$). We can show that such a set always exists.

Lemma 4. *Let $S = \{s_0, s_1, \ldots, s_z\}$ be any $(\alpha, \alpha+1)$-ruling set of $\hat{P}_i$ for $\alpha = 14\kappa_D \log^3 n$, there always exists a family of the terminal sets $\mathcal{T} = \{T_0, T_1, \ldots, T_z\}$ associated with S.*

Proof. The proof is constructive. Let $c = 6\kappa_D \log^3 n$. We take an arbitrary shortest path $Q = (s_j = u_0, u_1, u_2, \ldots, u_c)$ of length c in $P_i + H_i^1$, starting from $s_j \in S$. Because no two nodes in $N^+(V_{P_i}) \setminus V_{P_i}$ are adjacent in $P_i + H_i^1$, Q contains no two consecutive nodes, which are both in $N^+(V_{P_i}) \setminus V_{P_i}$; implying that at least half of the nodes in Q belong to V_{P_i}. Let $q' = (u'_0, u'_1, \ldots u'_{c'})$ be the subsequence of Q consisting of the nodes in V_{P_i}. Then, we define $T_j = \{u'_0, u'_3, \ldots, u'_{3\lfloor c'/3 \rfloor}\}$, which satisfies the three properties of terminal sets: It is easy to check that the first and second properties hold. In addition, one can show that $\mathrm{dist}_G(u'_x, u'_{x+a}) \geq 3$ (which is equivalent to $N^+(u'_x) \cap N^+(u'_{x+a}) = \emptyset$) holds for any $a \geq 3$, and $x \in [1, c'-a]$: Suppose that $\mathrm{dist}_G(u'_x, u'_{x+a}) \leq 2$ holds for some $a \geq 3$ and $x \in [1, c'-a]$, the distance between u'_x and u'_{x+a} implies $N^+(u'_x) \cap N^+(u'_{x+a}) \neq \emptyset$, and thus $\mathrm{dist}_{\hat{P}_i}(u'_x, u'_{x+a}) \leq 2$ holds. Then, bypassing the subpath from u'_x to u'_{x+a} in Q through a distance-two path, we obtain a path from s_j to u_c shorter than Q. This contradicts the fact that Q is the shortest path. □

The second property of terminal sets and the following lemma deduces that $\mathrm{dist}_{P_i+H_i}(s_0, s_j) = \tilde{O}(\kappa_D)$ holds for any $j \in [0, z]$.

Lemma 5. *Let $S = \{s_0, s_1, \ldots, s_z\}$ be any $(\alpha, \alpha+1)$-ruling set of $\hat{P}_i$ for $\alpha = 14\kappa_D \log^3 n$, and $\mathcal{T} = \{T_0, T_1, \ldots, T_z\}$ be a family of terminal sets associated with S. For any $j \in [0, z]$, there exist $u \in T_0$ and $v \in T_j$ such that $\mathrm{dist}_{P_i+H_i}(u, v) = O(1)$ holds.*

Proof. Because the distances s_0 and s_j are at least $14\kappa_D \log^3 n$, we have $N^+(T_0) \cap N^+(T_j) = \emptyset$. The proof is divided into the cases of $D = 3$ and $D = 4$. $(D = 3)$ Under the conditions of $N^+(T_0) \cap N^+(T_j) = \emptyset$ and $D = 3$, there exists a path of length exactly three from any node $a \in T_0$ to any node $b \in T_j$. Letting $e_{a,b}$ be the second edge in that path, we define $F = \{e_{a,b} \mid a \in T_0, b \in T_j\}$. By the third property of the terminal sets and because $N^+(T_0) \cap N^+(T_j) = \emptyset$, for any two edges $(x_1, y_1), (x_2, y_2) \in F$, either $x_1 \neq x_2$, or $y_1 \neq y_2$ holds; that is, $e_{a_1,b_1} \neq e_{a_2,b_2}$ holds for any $a_1, a_2 \in T_0$, $b_1, b_2 \in T_j$ and $(a_1, b_1) \neq (a_2, b_2)$. The first property of terminal sets implies that $|F| = |T_0||T_j| \geq (\kappa_D \log^3 n)^2$. Because each edge in F is added to H_i^2, with probability $1/n^{1/2} = 1/\kappa_D^2$, the probability that no edge in F is added to H_i^2 is at most $(1 - 1/\kappa_D^2)^{(\kappa_D \log^3 n)^2} \leq e^{-\Omega(\log^6 n)}$;

that is, an edge $e_{a,b}$ is added to H_i whp. and then $\mathrm{dist}_{P_i+H_i}(a,b) \leq 3$ holds. ($D=4$) For any node $u \in T_0$ and $v \in T_j$, there exists a path from u to v of length three or four in G. That path necessarily contains a length-two sub-path $P_2(u,v) = (a_{uv}, b_{uv}, c_{uv})$ such that $a_{uv} \in N^+(u)$ and $c_{uv} \in N^+(v)$ hold (if $P_2(u,v)$ is not uniquely determined, an arbitrary length-two sub-path is chosen). We refer to (a_{uv}, b_{uv}) and (b_{uv}, c_{uv}), the *first* and *second edges* of $P_2(u,v)$, respectively. (See Figure 2.3.) Let $\mathcal{P}_2 = \{P_2(u,v) \mid u \in T_0, v \in T_j\}$, G' is the union of $P_2(u,v)$ for all $u \in T_0$ and $v \in T_j$, and $\mathcal{P}_2^e = \{P_2(u,v) \in \mathcal{P}_2, \mid e \in P_2(u,v)\}$ for any $e \in E_{G'}$. We first bound the size of $\mathcal{P}_2^e$. Assume that e is the first edge of a path in $\mathcal{P}_2^e$. Let $e=(a,b)$ and $u \in T_0$ be the (unique) node such that $a \in N^+(u)$ holds. Because at most $|T_j|$ paths in $\mathcal{P}_2$ can start from a node in $N^+(u)$, the number of paths in $\mathcal{P}_2$ using e as their first edges is at most $|T_j|$. Similarly, if e is the second edge of some path in $\mathcal{P}_2^e$, at most $|T_0|$ paths in $\mathcal{P}_2$ can contain e as their second edges. Although some edges may be used as both the first and second edges, the total number of paths using e is bounded by $|T_0| + |T_j| = 2\kappa_D \log^3 n$, implying that any path $P_2(u,v)$ can share edges with at most $4\kappa_D \log^3 n$ edges, and thus, $\mathcal{P}_2$ contains at least $|T_0||T_j|/(4\kappa_D \log^3 n + 1) \geq \kappa_D \log^3 n/5$ edge-disjoint paths. Let $\mathcal{P}_2' \subseteq \mathcal{P}_2$ be the maximum-cardinality subset of $\mathcal{P}_2$ such that any $P_2(u_1,v_1), P_2(u_2,v_2) \in \mathcal{P}_2'$ is edge-disjoint. We define $B = \{b \mid (a,b,c) \in \mathcal{P}_2'\}$. Let $\Delta(b)$ be the number of paths in $\mathcal{P}_2'$, containing $b \in B$ as the center. Owing to the edge disjointness of $\mathcal{P}_2'$, we have $|E_G(N^+(T_0), b)| \geq \Delta(b)$, and $|E_G(N^+(T_j), b)| \geq \Delta(b)$ for any $b \in B$. Let Y_b be the value of $h(b,i)$, and X_b be the indicator random variable that takes one if a path in $\mathcal{P}_2$, which contains b as the center, is added to H_i, and zero otherwise. Let X and Y be the indicator random variables corresponding to the events of $\bigvee_{b\in B}(X_b = 1)$ and $\bigvee_{b\in B}(Y_b \leq \Delta(b)/\log^2 n)$ respectively. By the definition of $\Delta(b)$, the probability of adding no first edge with b and the node in T_0 as the endpoints to H_i is at most $(1-1/h(b,i))^{\Delta(b)}$. Similarly, the probability of adding no second edge with b and the node in T_j as the endpoints to H_i is at most $(1-1/h(b,i))^{\Delta(b)}$. Then, we obtain $\Pr[X_b = 1 \mid Y_b = y] \geq 1 - (1-1/y)^{\Delta(b)} - (1-1/y)^{\Delta(b)} \geq 1 - 2e^{-\Delta(b)/y}$, and thus, $\Pr[X_b = 1 \mid Y_b \leq \max\{1, \Delta(b)/\log^2 n\}] \geq 1 - e^{-\Omega(\log^2 n)}$ holds. Note that, if $Y_b = 1$, then all incident edges of b included in $\mathcal{P}_2'$ must be added to H_i, so $X_b = 1$ always holds. Therefore, $\Pr[X = 1 \mid Y = 1] \geq 1 - e^{-\Omega(\log^2 n)}$ holds. Because h is $(n^{1/3}\log^3 n)$-wise independent, Y_b for all $b \in B$ are independent. Thus, we obtain the following:

$$
\begin{aligned}
\Pr[Y=1] &= 1 - \Pr[Y=0] \\
&= 1 - \Pr\left[\bigwedge_{b\in B} Y_b > \max\left\{1, \frac{\Delta(b)}{\log^2 n}\right\}\right] \\
&= 1 - \prod_{b\in B, \Delta(b) > 2\log^2 n} \Pr\left[Y_b > \frac{\Delta(b)}{\log^2 n}\right] \\
&\quad \times \prod_{b\in B, \Delta(b) \leq 2\log^2 n} \Pr\left[Y_b > 1\right], \\
&= 1 - \prod_{b\in B, \Delta(b) > 2\log^2 n} \left(1 - \left\lfloor \frac{\Delta(b)}{n^{\frac{1}{3}}\log n} \right\rfloor\right) \\
&\quad \times \prod_{b\in B, \Delta(b) \leq 2\log^2 n} \left(1 - \frac{\log n}{n^{\frac{1}{3}}}\right), \\
&= 1 - \prod_{b\in B, \Delta(b) > 2\log^2 n} \left(1 - \frac{\Delta(b)}{2n^{\frac{1}{3}}\log n}\right)
\end{aligned}
$$

$$
\begin{aligned}
&\times \prod_{b\in B,\Delta(b)\leq 2\log^2 n}\left(1-\frac{2\log^2 n}{2n^{\frac{1}{3}}\log n}\right),\\
&\geq 1-exp\left(-\sum_{b\in B,\Delta(b)>2\log^2 n}\frac{\Delta(b)}{2n^{\frac{1}{3}}\log n}\right)\\
&\quad\times exp\left(-\sum_{b\in B,\Delta(b)\leq 2\log^2 n}\frac{2\log^2 n}{2n^{\frac{1}{3}}\log n}\right),\\
&\geq 1-exp\left(-\sum_{b\in B,\Delta(b)>2\log^2 n}\frac{\Delta(b)}{2n^{\frac{1}{3}}\log n}\right)\\
&\quad\times exp\left(-\sum_{b\in B,\Delta(b)\leq 2\log^2 n}\frac{\Delta(b)}{2n^{\frac{1}{3}}\log n}\right),\\
&= 1-exp\left(-\sum_{b\in B}\frac{\Delta(b)}{2n^{\frac{1}{3}}\log n}\right)\\
&\geq 1-exp\left(-\frac{|\mathcal{P}_2'|}{2n^{\frac{1}{3}}\log n}\right),\\
&\geq 1-exp(-\Omega(\log^2 n)).
\end{aligned}
$$

Consequently, we have that $\Pr[X=1]\geq \Pr[X=1\wedge Y=1]\Pr[Y=1]\geq \left(1-e^{-\Omega(\log n)}\right)^2$. The lemma is proved. □

2.4.2 Distributed Implementation

We explain the above-mentioned implementation details of the algorithm in the CONGEST model as follows:

- (**Preprocessing**) In the algorithm stated above, the shortcut construction is performed only for large parts, which is crucial to bound the congestion of each edge. Thus, as a preprocessing task, each node has to know if its own part is large (i.e., having a diameter larger than κ_D) or not. The exact identification of the diameter is usually a hard task; only an asymptotic identification is sufficient for achieving the shortcut quality stated above, where the parts of diameter $\omega(\kappa_D)$ and diameter $o(\kappa_D)$ must be identified as large and small ones, but those of diameter $\Theta(\kappa_D)$ are arbitrarily identified. This loose identification is easily implemented through simple distance-bounded aggregation. The algorithm for part P_i is as follows: (1) In the first round, each node in P_i sends its ID to all the neighbors, and (2) in the subsequent rounds, each node forwards the minimum ID received thus far. The algorithm executes this message propagation during κ_D rounds. If the diameter is substantially larger than κ_D, the minimum ID in P_i does not reach all the nodes in P_i. Then, there exists an edge whose endpoints identify different minimum IDs. The one-more-round propagation allows those endpoints to know the part is large. Thereafter, they start to broadcast the signal "large" using the following κ_D rounds. If κ_D is large, the signal "large" is invoked at several nodes in P_i, and κ_D-round propagation guarantees that every node receives the signal. That is, any node in P_i identifies that P_i is large. The running time of this task is $O(\kappa_D)$ rounds.

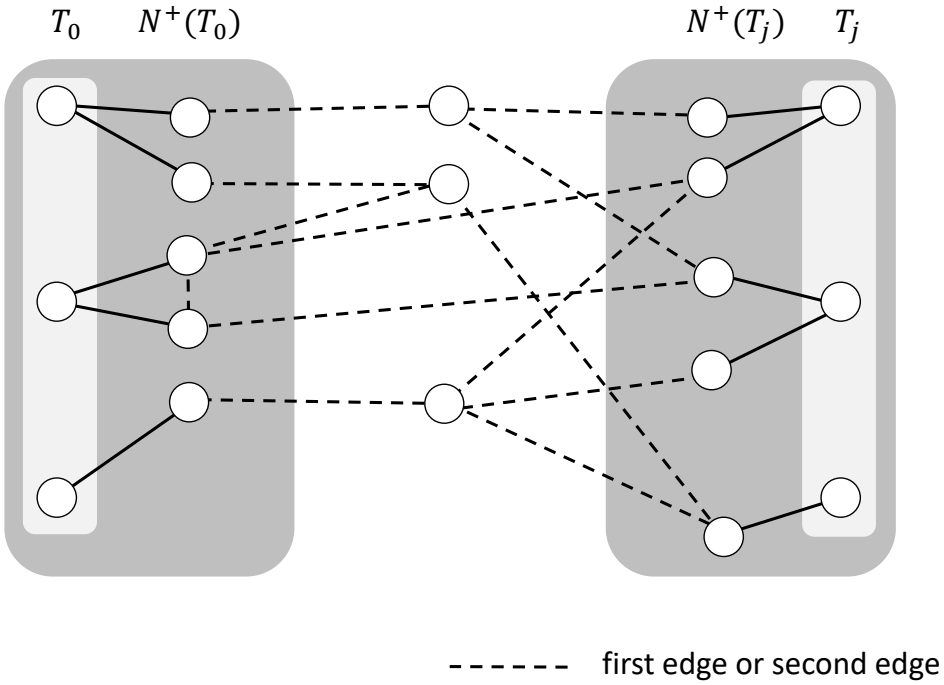

Figure 2.3: Proof of Lemma 5.

- (**Step 1**) As we stated, the 1-hop extension is implemented in one round. In this step, each node $v \in V_{P_i}$ tells all the neighbors if P_i is large or not. Consequently, if part P_i is identified as a large one, all the nodes in $N^+(P_i)$ know it after this step.
- (**Step 2**) The algorithm for $D = 3$ is trivial. For $D = 4$, there are two non-trivial matters. The first matter is the preparation of hash function h. We realize it by sharing a random seed of $O(n^{1/3} \log^3 n \log |\mathcal{Y}|)$-bit length in advance. A standard construction by Wegman and Carter [79] allows each node to construct the desired h in common. Sharing the random seed is implemented by the broadcast of one $O(n^{1/3} \log^3 n \log |\mathcal{Y}|)$-bit message, i.e., taking $\tilde{O}(\kappa_D)$ rounds. The second matter is to address that u does not know whether P_i is large or not, and/or if v belongs to $N^+(P_i)$. It makes u difficult to determine if (u, v) should be added to H_i. Instead, our algorithm simulates the task of u by the nodes in $N(u)$. More precisely, each node $v \in N^+(V_{P_i})$ adds each incident edge (u, v) to H_i with probability $1/h(u, i)$. Because $v \in N^+(P_i)$, v knows if P_i is large or not (informed in step 1), and v can also compute $h(u, i)$ locally. Thus, the choice of (u, v) is locally decidable at v. Because this simulation is completely equivalent to the centralized version, the analysis of the quality also applies.

It is easy to check that the construction time of the distributed implementation above is $\tilde{O}(\kappa_D)$ in total.

2.5 Low-Congestion Shortcut for Bounded Clique-width Graphs

Let $G = (V(G), E(G))$ be a graph. A k-graph ($k \geq 1$) is a graph whose vertices are labeled by integers in $[1, k]$. A k-graph is naturally defined as a triple $(V(G), E(G), f)$, where f is the labeling function $f : V \to [1, k]$. The clique width $G = (V(G), E(G))$ is the minimum k such that there exists a k-graph $G = (V(G), E(G), f)$, which is constructed through repeated application of the

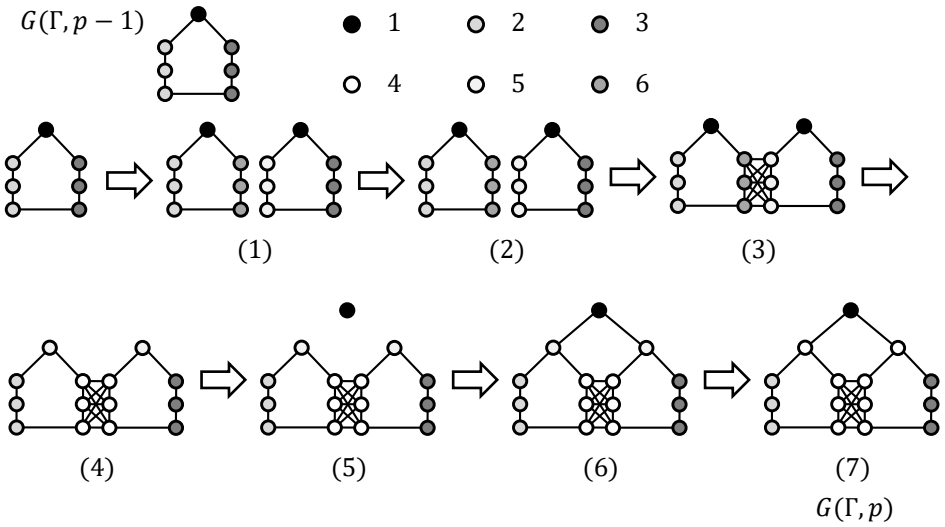

Figure 2.4: Graph $G \oplus H$.

following four operations: (1) introduce: create a graph of a single node v with label $i \in [1, k]$, (2) disjoint union: take the union $G \cup H$ of two k-graphs G and H, (3) relabel: given $i, j \in [1, k]$, change all the labels i in the graph to j, and (4) join: given $i, j \in [1, k]$, connect all vertices labeled by i, with all vertices labeled by j by edges.

The clique-width is invented first as a parameter to capture the tractability for an easy subclass of high treewidth graphs [16, 19]; that is, the class of bounded clique-width can contain several graphs with high treewidths. In centralized settings, one can often obtain polynomial-time algorithms for several non-deterministic-polynomial complete (NP-complete) problems under the assumption of bounded clique-width. Courcelle et al. [18] showed that for some fixed k, any problem that can be expressed in Monadic second-order logic with quantification over vertices (MSO_1) can be solved in linear time on any class of graphs of clique-width at most k if we obtain k-expressions that define the input graph. Coudert et al. [17] showed several polynomial dependencies in the fixed parameter algorithm (P-FPT) in a bounded clique-width graph. The following negative result, however, states that bounding clique-width does not admit any good solution for the MST problem (and thus, for the low-congestion shortcut).

Theorem 6. *There exists an unweighted n-vertex graph $G = (V(G), E(G))$ of clique-width six, where for any MST algorithm A, there exists an edge-weight function $w_A : E(G) \to \mathbb{N}$ such that the running time of A becomes $\tilde{\Omega}(\sqrt{n} + D)$ rounds.*

We introduce the instance stated in this theorem, which is denoted by $G(\Gamma, p)$ (Γ and p are the parameters fixed later), using the operations specified in the definition of clique width; that is, this introduction itself becomes the proof of clique-width six. Let $\mathcal{G}(\Gamma)$ be the set of 6-graphs that contains one node with label 1, Γ nodes with label 2, and Γ nodes with label 3, and all other nodes are labeled by 4. Then, we define the binary operation $\oplus$ over $\mathcal{G}(\Gamma)$. For any $G, H \in \mathcal{G}(\Gamma)$, the graph $G \oplus H$ is defined as the graph obtained through the following operations: (1) Relabel 2 in G with 5 and relabel 3 in H with 6, (2) take the disjoint union $G \cup H$, (3) join with labels 5 and 6, (4) relabel 5 and 6 with 4, and then 1 with 5, (5) add a node with label 1 through operation introduce, (6) join with 1 and 5, and (7) relabel 5 with 4. This process is illustrated in Figure 2.4.

Now we are ready to define $G(\Gamma, p)$. The construction is recursive. First, we define $G(\Gamma, 1)$ as follows: (1) Prepare a (2Γ)-biclique $K_{\Gamma,\Gamma}$, where one side has label 2, and the other side has

label 3. Notably, two labels suffice to construct $K_{\Gamma,\Gamma}$. (2) Add three nodes with labels 1, 5, and 6 through operation introduce. (3) Join with labels 2 and 5, and with 3 and 6. (4) Join with label 1 and 5, and with 1 and 6. (5) Relabel 5 and 6 with 4. Thereafter, we define $G(\Gamma, p) = G(\Gamma, p-1) \oplus G(\Gamma, p-1)$. The instance claimed in Theorem 6 is $G(\sqrt{n}, \log n/2)$, which is illustrated in Figure 2.5. This instance is very close to the standard hard-core instance used in prior work (for example, [75,76]. See Figure 2.1). Thus it is not difficult to observe that $\tilde{\Omega}(\sqrt{n})$-round lower bound for MST construction also applies to $G(\sqrt{n}, \log n/2)$. It suffices to show the subsequent lemma. Combined with Theorem 3, we obtain Theorem 6.

Lemma 6. $G(\Gamma, p) \in \mathcal{G}(O(\Gamma(2^p+2)), \Gamma, 2^p+2, 3p)$.

Proof. First, let us formally specify the graph $G(\Gamma, p)$, which is defined as follows (vertex IDs introduced below are described in Figure 2.5):

- $V(\Gamma, p) = T \cup \bigcup_{1 \leq l \leq \Gamma} V_l$ such that $T = \{u_i^j \mid 0 \leq i \leq 2^j - 1, 0 \leq j \leq p\}$, $V_l = \{v_i^l \mid 0 \leq i \leq 2^p - 1\}$.
- $E(\Gamma, p) = E_1 \cup E_2 \cup E_3$ such that $E_1 = \{(u_i^j, u_{\lfloor \frac{i}{2} \rfloor}^{j-1}) \mid 0 \leq i \leq 2^j - 1, 1 \leq j \leq p\}$. $E_2 = \{(u_i^p, v_i^j) \mid 0 \leq i \leq 2^p - 1, 1 \leq j \leq \Gamma\}$, $E_3 = \{(v_i^j, v_{i+1}^k) \mid 0 \leq i \leq 2^p - 2, 1 \leq j \leq \Gamma, 1 \leq k \leq \Gamma\}$.

We define $\mathcal{X} = \{X_1, X_2, \ldots, X_{2^p+2}\}$ for graph $G(\Gamma, p)$ as follows:

- $X_i = \{u_0^p\}$ $(i = 1)$.
- $X_i = \{v_0^j \mid 1 \leq j \leq \Gamma\}$ $(i = 2)$.
- $X_i = \{v_{i-2}^j \mid 2 \leq j \leq N\} \cup \{u_{(i-1)/2^j - 1}^{p-j} \mid 0 \leq j \leq p, i - 1 \bmod 2^j = 0\}$ $(3 \leq i \leq 2^p - 1)$.
- $X_i = \{v_{2^p-2}^j \mid 1 \leq j \leq \Gamma\} \cup \{u_{2^p-2}^p\} \cup \{u_{2^j-1}^j \mid 0 \leq j \leq p-1\}$ $(i = 2^p)$.
- $X_i = \{v_{2^p-1}^j \mid 1 \leq j \leq \Gamma\}$ $(i = 2^p + 1)$
- $X_i = \{u_{2^p-1}^p\}$ $(i = 2^p + 2)$

We define $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_\Gamma\}$ for graph $G(\Gamma, p)$ as follows:

- $Q_i = V_1 \cup (T \backslash (s \cup r))$ $((i = 1))$.
- $Q_i = V_i$ $((2 \leq i \leq \Gamma))$.

It is easy to check that (**C1**) and (**C2**) are satisfied. Thus, we only show that (**C3**) is satisfied. Let $V_{R_i} = R_i \cap \bigcup_{j=1}^{\Gamma} V_j$. For $2 \leq i \leq (2^p+2)/2$, we have $(N(V_{R_i}) \backslash R_{i-1}) = \emptyset$. For any ℓ and $1 \leq i \leq 2^{p-2}$, if u_i^p is included in R_ℓ, then the neighbors of u_i^p are included in R_ℓ. For any ℓ, $1 \leq i \leq p$ and $0 \leq j \leq 2^i - 2$, if u_j^i is included in R_ℓ, then u_{j+1}^i is included in R_ℓ. Let $u^i(R_\ell)$ be the leftmost vertex whose level is i of T and included in R_ℓ. For any ℓ, $1 \leq i \leq p$ and $0 \leq j \leq 2^i - 1$, if $u_j^i \neq u^i(R_\ell)$ and u_j^i are included in R_ℓ, then the parent of u_j^i is included in R_ℓ. Thus, $|(N(R_\ell) \backslash R_{\ell-1})|$ only includes neighbors of $u^i(R_\ell)$ for $1 \leq i \leq p$ and $2 \leq \ell \leq (2^p+2)/2$. Because the tree T is a binary tree, $u^i(R_\ell)$ has at most 3 neighbors in T. Therefore we have $|E((N(R_i) \backslash R_{i-1}))| \leq 3p$. Similarly, we have $|E((N(L_i) \backslash L_{i-1}))| \leq 3p$. Therefore, we can prove that the graph $G(\Gamma, p)$ is included in $\mathcal{G}(O(\Gamma(2^p+2)), \Gamma, 2^p+2, 3p)$. By Theorem 3, the lower bound for constructing the MST in $\mathcal{G}(O(\Gamma(2^p+2)), \Gamma, 2^p+2, 3p)$ is $\tilde{\Omega}((\min\{\Gamma/3p, ((2^p+2)/2 - 1\})$. When $\Gamma = \Theta(\sqrt{n})$ and $2^p = \Theta(\sqrt{n})$, we obtain the $\tilde{\Omega}(\sqrt{n})$ lower bound. □

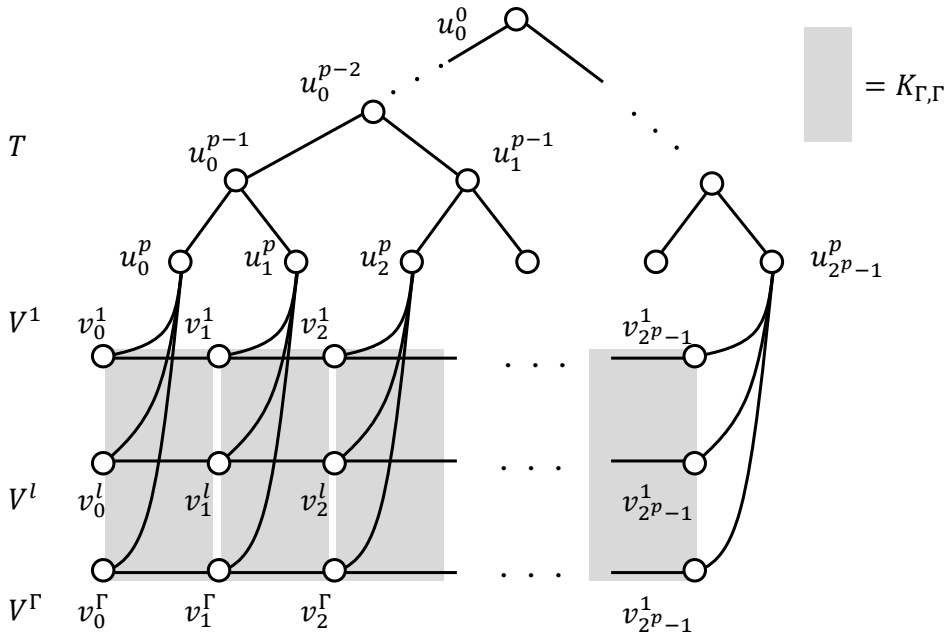

Figure 2.5: Example of clique-width 6 graph $G(\Gamma, p)$.

Chapter 3

A Subquadratic-Time Distributed Algorithm for Exact Maximum Matching

3.1 Introduction

In this chapter, we consider the maximum matching problem in the CONGEST model. We present the main theorem of this chapter below.

Theorem 7. *For any input graph G, there exists a randomized CONGEST algorithm to compute the maximum matching that terminates within $O\left(s_{\max}^{3/2}\right)$ rounds with probability $1 - 1/n^{\Theta(1)}$.*

3.2 Preliminaries

We use the definition of the CONGEST model defined in Section 2.2.1.

3.2.1 Matching and Augmenting Path

For a graph $G = (V(G), E(G))$, a matching $M \subseteq E(G)$ is a set of edges that do not share endpoints. A node v is called a *matched node* if M intersects $I_G(v)$, or an *unmatched node* otherwise. A path $U = \{v_0, e_0, v_1, e_1, \ldots, v_\ell\}$ is called an *alternating path* if $\mathbf{I}_M(e_i) + \mathbf{I}_M(e_{i+1}) = 1$ holds for any $1 \leq i \leq \ell - 1$.[1] If the length $|U|$ of U satisfies $|U| \mod 2 = \theta$, U is called *θ-alternating*. The value θ is called the *parity* of U. By definition, any 0-alternating (1-alternating) path from an unmatched node f finishes with a matching (non-matching) edge. Due to a technical issue, we regard the path of length zero as a 0-alternating path. For any $\theta \in \{0, 1\}$ and $u, v \in V(G)$, we define $r^\theta(u, v)$ as the length of the shortest θ-alternating path between u and v. An *augmenting path* is an alternating path connecting two unmatched nodes. Note that the augmenting path must be 1-alternating path. We say that (G, M) has an augmenting path if there exists an augmenting path in G with respect to M. The following proposition is a well-known fact in the maximum matching problem.

Proposition 2. *Given a matching $M \subseteq E(G)$ of graph G, M is the maximum matching if and only if (G, M) has no augmenting path.*

[1] The indicator function $\mathbf{I}_X(x)$ returns one if $x \in X$ and zero otherwise.

3.2.2 Approximate Maximum Matching

Our algorithm uses an $O(1)$-approximate upper bound for the maximum matching size of the input graph. To obtain the upper bound, we run the $O(s_{\max})$-round maximal matching algorithm as follows. First we suppose each edge has a unique ID and priority associated with the ID. We run a simple parallel greedy algorithm, where each node adds an edge to the matching if all neighboring higher priority edges are already known not be in the maximal matching. One iteration of the algorithm increases the matching size at least by one, and thus $O(s_{\max})$ rounds suffices to obtain a maximal matching. Since $s_{max} = \Omega(D(G))$ always holds, the termination of maximal matching construction is detected in $O(s_{\max})$ rounds by checking whether all nodes terminated or not. Note that the termination detection is executed once in every $\Theta(D(G))$ rounds. Let M^* be the computed maximal matching. Since any maximal matching is a $(1/2)$-approximate maximum matching, one can obtain the bound $2|M^*| \geq s_{\max}$. The size $s_{\max}$ is at least half of the diameter $D(G)$, and thus we can spend $O(D(G)) = O(s_{\max})$ rounds for counting and propagating the number of edges in M^*. That is, it is possible to provide each node with the value of $2|M^*|$ in a preprocessing phase using only $O(D(G)) = O(s_{\max})$ rounds. Let $s^*_{\max}$ be the number of maximal matching. In the following argument, we denote $\hat{s}_{\max} = 2s^*_{\max}$, the value of which is available to each node.

3.2.3 Maximum-Matching Verification Algorithm

Our algorithm uses the algorithm by Ahmadi et al.'s [5] for maximum-matching verification as a building block. Although the original algorithm is designed for the verification of maximum matching, it provides each node with information on the length of alternating paths to the closest unmatched nodes. Precisely, the following lemma holds.

Theorem 8 (Ahmadi et al. [5])**.** *Assume that a graph $G = (V(G), E(G))$ and a matching $M \subseteq E(G)$ are given, and let W be the set of all unmatched nodes. There exist two $O(\ell)$-round randomized CONGEST algorithms $\mathsf{MV}(M, \ell, f)$ and $\mathsf{PART}(M, \ell)$ that output the following information at every node $v \in V(G)$ with a probability of at least $1 - 1/n^c$ for an arbitrarily large constant $c > 1$.*

1. *Given M, a nonnegative integer ℓ, and a node $f \in W$, $\mathsf{MV}(M, \ell, f)$ outputs the pair $(\theta, r^\theta(f, v))$ at each node v if $r^\theta(f, v) \leq \ell$ holds (if the condition is satisfied for both $\theta = 0$ and $\theta = 1$, v outputs two pairs). The algorithm $\mathsf{MV}(M, \ell, f)$ is initiated only by the node f (with the value ℓ), and other nodes do not require information on the ID of f and value ℓ at the initial stage.*

2. *The algorithm $\mathsf{PART}(M, \ell)$ outputs a partition $V^1, V^2, \ldots, V^N$ of $V(G)$ (as the label i for each node in V^i) such that (a) For $1 \leq i \leq N - 1$, the subgraph G^i induced by V^i contains exactly two unmatched nodes f^i and g^i as well as an augmenting path between f^i and g^i of length at most ℓ and (b) the diameter of G^i is $O(\ell)$. If G contains an augmenting path, $\mathsf{PART}(M, \ell)$ always returns at least two sets of vertices, otherwise $\mathsf{PART}(M, \ell)$ returns the set of vertices, that is, $V^N = V(G)$. Note that, $\mathsf{PART}(M, \ell)$ can be applied in a subgraph.*

While the original paper [5] presents a single algorithm returning the outputs of both MV and PART, we intentionally separate it into two algorithms with different roles for clarity. Note that our matching-construction algorithm uses random bits only in the runs of these algorithms. As our algorithm uses them only $O(\mathrm{poly}(n))$ times as subroutines, we can guarantee that our algorithm has a high probability of success by taking a sufficiently large c. Hence, we do not pay much attention to the failure probability of our algorithm. Any stochastic statement in the following argument also holds with probability $1 - n^c$ for an arbitrary constant $c > 1$.

Algorithm 1 Constructing a maximum matching in $O(n^{3/2})$ rounds.

1: **for** $i = 1; i \leq \hat{s}_{\max} - \sqrt{\hat{s}_{\max}}; i++$ **do**
2: run the algorithm $\mathsf{A}(M, \ell)$ with $\ell = \lceil 2\hat{s}_{\max}/(\hat{s}_{\max} - i) \rceil$ for $O(\ell)$ rounds.
3: **if** $\mathsf{A}(M, \ell)$ finds a nonempty set of vertex-disjoint augmenting paths within $O(\ell)$ rounds, **then**
4: improve the current matching using the set of vertex-disjoint augmenting paths.
5: **for** $i = 1; i \leq \sqrt{\hat{s}_{\max}}; i++$ **do**
6: run the algorithm $\mathsf{B}(M)$ for $O(\hat{s}_{\max})$ rounds.
7: **if** $\mathsf{B}(M)$ finds a nonempty set of vertex-disjoint augmenting paths within $O(\hat{s}_{\max})$ rounds, **then**
8: improve the current matching M using the set of vertex-disjoint augmenting paths.

3.3 Computing the Maximum Matching in CONGEST

As explained in the introduction, the maximum matching problem is reducible to the problem of finding an augmenting path. We first present two key results below.

Lemma 7. *Let M be a matching of G. Provided that (G, M) has exactly two unmatched nodes $f, g \in V_G$ and contains an augmenting path of length at most ℓ between f and g, there exists an $O(\ell^2)$-round randomized algorithm that outputs an augmenting path connecting f and g.*

Lemma 8. *Let M be a matching of G. Provided that (G, M) has exactly two unmatched nodes $f, g \in V_G$ and contains an augmenting path between f and g, there exists an $O(n)$-round randomized algorithm that outputs an augmenting path that includes f.*

The outputs of both algorithms are the labels to the edges in the computed augmented path. If the edge e is included in the augmenting path, then the vertices connecting to e know that e is included in the augmenting path. Each node adds the edge e to a matching M if the edge e is included in the augmenting path and is not included in the matching, and removes the edge e from a matching M if the edge e is included in the augmenting path and the matching. To prove the lemmas, one can utilize the output of the algorithm PART. We first run the verification algorithm $\mathsf{PART}(M, \ell)$ (for Lemma 7) or $\mathsf{PART}(M, \hat{s}_{\max})$ (for Lemma 8) as a preprocessing step and then execute the algorithms of Lemma 7 or 8 for each G^i output by PART independently. Note that each G^i contains only matched nodes and two unmatched nodes; thus, $|V(G^i)| \leq 2|M| + 2$ holds for any G^i. Then, the following corollary is deduced:

Corollary 1. *There exist two randomized algorithms $\mathsf{A}(M, \ell)$ and $\mathsf{B}(M)$ satisfying the following conditions, respectively:*

- *For any graph $G = (V(G), E(G))$ and matching $M \subseteq E(G)$, $\mathsf{A}(M, \ell)$ finds a nonempty set of vertex-disjoint augmenting paths within $O(\ell^2)$ rounds if (G, M) has an augmenting path of length at most ℓ.*
- *For any graph $G = (V(G), E(G))$ and matching $M \subseteq E(G)$, $\mathsf{B}(M)$ finds a nonempty set of vertex-disjoint augmenting paths of (G, M) within $O(|M|)$ rounds if (G, M) has an augmenting path.*

We present an $O\left(s_{\max}^{3/2}\right)$-round algorithm for computing the maximum matching using the algorithms $\mathsf{A}(M, \ell)$ and $\mathsf{B}(M)$. The pseudocode of the whole algorithm is presented in Algorithm 1.

It basically follows the standard idea of centralized maximum matching algorithms, i.e., finding an augmenting path and improving the current matching iteratively. The first $\hat{s}_{\max} - \sqrt{\hat{s}_{\max}}$ iterations use $\mathsf{A}(M, \ell)$ (lines 1–4), and the remaining $\sqrt{\hat{s}_{\max}}$ iterations use $\mathsf{B}(M)$. In the i-th iteration, the algorithm $\mathsf{A}(M, \ell)$ runs with $\ell = \lceil 2\hat{s}_{\max}/(2\hat{s}_{\max} - i) \rceil$. This setting comes from Proposition 1. The improvement of the current matching by a given augmenting path is simply a local operation and is realized by flipping the labels of matching edges and non-matching edges on the path. The correctness and running time of Algorithm 1 are analyzed below.

Lemma 9. *Algorithm 1 constructs a maximum matching with high probability in $O\left(s_{\max}^{3/2}\right)$ rounds.*

Proof. Let $\xi(i)$ be the matching size at the end of i iterations of the algorithm $\mathsf{A}(M, \ell)$. We show that $\xi(\hat{s}_{\max} - s_{\max} + j) \geq j$ holds for any $0 \leq j \leq s_{\max} - \sqrt{\hat{s}_{\max}}$. It implies that the matching size is at least $s_{\max} - \sqrt{\hat{s}_{\max}}$ after the application of $\mathsf{A}(M, \cdot)$. Therefore, the maximum matching is constructed by $\sqrt{\hat{s}_{\max}}$ iterations of the algorithm $\mathsf{B}(M)$. The proof of the statement above follows the induction on j. (Basis) If $j = 0$, the statement trivially holds. (Inductive step) As the induction hypothesis, suppose $\xi(\hat{s}_{\max} - s_{\max} + j') \geq j'$ holds. If $\xi(\hat{s}_{\max} - s_{\max} + j') \geq j' + 1$, then the statement holds. Therefore, we consider the case in which $\xi(\hat{s}_{\max} - s_{\max} + j') = j'$ holds. By Proposition 1, there exists an augmenting path of length at most $\lfloor 2s_{\max}/(s_{\max} - j') \rfloor \leq 2\hat{s}_{\max}/(s_{\max} - j') = 2\hat{s}_{\max}/(\hat{s}_{\max} - (\hat{s}_{\max} - s_{\max} + j')) \leq 2\hat{s}_{\max}/(\hat{s}_{\max} - (\hat{s}_{\max} - s_{\max} + (j' + 1)))$ at the end of $\hat{s}_{\max} - s_{\max} + j'$ iterations of the algorithm $\mathsf{A}(M, \ell)$. Hence, the size of the matching is increased by at least one in the $(\hat{s}_{\max} - s_{\max} + j' + 1)$-th iteration.

Now, we show the running-time analysis of Algorithm 1. Recall that $\hat{s} = \Theta(s_{max})$ holds. As $\mathsf{A}(M, \ell)$ is repeated $\hat{s}_{\max} - \sqrt{\hat{s}_{\max}}$ times and $\mathsf{B}(M)$ is repeated $\sqrt{\hat{s}_{\max}}$ times, the running time of Algorithm 1 is as follows.

$$
\begin{aligned}
& O\left(s_{\max}\right) + O\left(\sum_{i=1}^{\hat{s}_{\max} - \sqrt{\hat{s}_{\max}}} \left(\left\lceil \frac{2\hat{s}_{\max}}{\hat{s}_{\max} - i} \right\rceil \right)^2 \right) + O\left(\hat{s}_{\max}\sqrt{\hat{s}_{\max}}\right) \\
=& O\left(\sum_{i=1}^{\hat{s}_{\max} - \sqrt{\hat{s}_{\max}}} \left(\frac{\hat{s}_{\max}}{\hat{s}_{\max} - i} \right)^2 + \hat{s}_{\max} - \sqrt{\hat{s}_{\max}} + \hat{s}_{\max}\sqrt{\hat{s}_{\max}} \right) \\
=& O\left(\sum_{i=\sqrt{\hat{s}_{\max}}}^{\hat{s}_{\max} - 1} \left(\frac{\hat{s}_{\max}}{i} \right)^2 + \hat{s}_{\max}\sqrt{\hat{s}_{\max}} \right) \\
=& O\left(\hat{s}_{\max}^2 \sum_{i=\sqrt{\hat{s}_{\max}}}^{\hat{s}_{\max} - 1} \left(\frac{1}{i} \right)^2 + \hat{s}_{\max}\sqrt{\hat{s}_{\max}} \right) \\
=& O\left(\hat{s}_{\max}^2 \frac{1}{\sqrt{\hat{s}_{\max}}} + \hat{s}_{\max}\sqrt{\hat{s}_{\max}} \right) \\
=& O\left(\hat{s}_{\max}^{3/2}\right) \\
=& O\left(s_{\max}^{3/2}\right).
\end{aligned}
$$

□

The following sections are devoted to proving Lemmas 7 and 8. Since the presented algorithms are intended to run in each G^i returned by the preprocessing run of $\mathsf{PART}(M, \cdot)$, without loss of

Algorithm 2 Construction of the augmenting path $\mathrm{CAP}((G, M), f, g, \ell)$.

Require: The path U_0 is an augmenting path with length ℓ from f to g.

$U_0, U_1, \ldots, U_\ell$: initially $\emptyset$.
$\mathsf{target} = g$
for $i = 1; i \leq \ell; i++$ **do**
  if i is even **then**
    target chooses the node $v_{\ell-i}$ that satisfies $\mathbf{I}_M((\mathsf{target}, v_{\ell-i})) = 1$.
    $U_{\ell-i} \leftarrow U_{\ell-i+1} \cup \{(\mathsf{target}, v_{\ell-i})\}$.
    $\mathsf{target} \leftarrow v_{\ell-i}$.
  else
    run the algorithm $\mathsf{MV}(M, \ell - i, f)$ with the subgraph $H_{\ell-i+1}$ induced by $V(G - U_{\ell-i+1})$ as the input.
    for any $v \in V(G - U_{\ell-i+1})$, the node v sends $r^0_{H_{\ell-i+1}}(f, v)$ to its neighborhood.
    target chooses a node $v_{\ell-i}$ that satisfies $\mathbf{I}_M((\mathsf{target}, v_{\ell-i})) = 0$ and $r^0_{H_{\ell-i+1}}(f, v_{\ell-i}) = \ell - i$.
    $U_{\ell-i} \leftarrow U_{\ell-i+1} \cup \{(\mathsf{target}, v_{\ell-i})\}$.
    $\mathsf{target} \leftarrow v_{\ell-i}$.

generality, we assume that G has exactly two unmatched nodes f and g with an augmenting path between them. In addition, it is assumed that one of f and g is elected as a primary unmatched node (referred to as f hereafter). This election process is easily implemented in $O(\ell)$ rounds because the distance between f and g is at most ℓ. When we argue the existence of augmenting or alternating paths in a subgraph $H = (V(H), E(H))$ of G, the matching $M \cap E(H)$ of graph H is considered without explicit notice. Given a subgraph $H \subseteq G$, we denote the length of the shortest odd (even) alternating path from f to v in H by $r^1_H(f, v)$ ($r^0_H(f, v)$). If no odd or even alternating path exists from f to v in H, then we define $r^1_H(f, v) = \infty$ or $r^0_H(f, v) = \infty$. As sentinels, we also define $r^0_H(f, f)$ as ∞ and $r^1_H(f, f)$ as 0.

3.4 Construction of Augmenting Path in $O(\ell^2)$ Rounds

3.4.1 Outline

Let $U = \{v_0, e_1, v_1, \ldots, v_\ell\}$ be the shortest augmenting path from f to g (i.e., $f = v_0$ and $g = v_\ell$) and $U_i = U^s_{v_i}$ for short. The key idea of the algorithm is to find the predecessor of each node v_i along U sequentially. Note that it does not suffice to choose a neighbor v of v_i with $r^\theta_G(f, v) = i - 1$ and $\mathbf{I}_M(v_i, v) = \theta$ for $\theta = (i - 1) \mod 2$ as the predecessor. This strategy is problematic in the scenario in which there exists two neighbors v and u such that $r^\theta_G(f, v) = r^\theta_G(f, u) = i - 1$ and $\mathbf{I}_M(v_i, u) = \mathbf{I}_M(v_i, v) = \theta$ for $\theta = (i - 1) \mod 2$, where u is the correct successor. While v is guaranteed to have the alternating path Q from f to v of length $i - 1$, it can intersect U_i. Then, the concatenation $Q \circ (v_i, v) \circ U_i$ is not simple. That is, it is not an augmenting path. To avoid this scenario, the algorithm finds the predecessor of v_i in the graph $G - U_i$, where $G - U_i$ is the induced graph by $V(G) \backslash V(U_i)$. If some neighbor v of v_i satisfies $r^\theta_{G-U_i}(f, v) = i - 1$ and $\mathbf{I}_M(v_i, v) = 1 - \theta$, the concatenated walk $Q \circ (v_i, v) \circ U_i$ is guaranteed to be simple.

3.4.2 Algorithm Details

Algorithm 2 details the algorithm for constructing the augmenting path in $O(\ell^2)$ rounds. The algorithm consists of ℓ steps. In the i-th step, it finds the predecessor of $v_{\ell-i+1}$. Assume that the

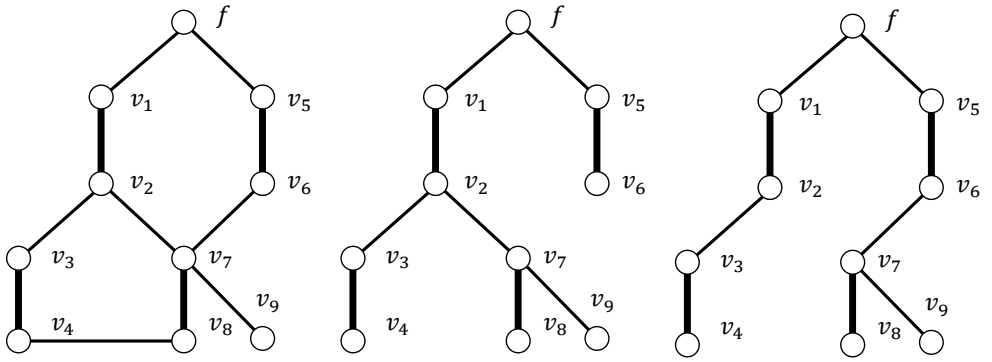

Figure 3.1: Examples of the alternating base tree. Bold lines are matching edges, and thin lines are unmatched edges.

algorithm has already found $U_{\ell-i+1}$ at the beginning of the i-th step. Any node in $V(U_{\ell-i+1}) \setminus \{v_{\ell-i+1}\}$ quits the algorithm (with the information of the predecessor in U_i), and thus, the nodes still running the algorithm are given by $V(G - U_{\ell-i+1})$. If i is even, the edge $(v_{\ell-i}, v_{\ell-i+1})$ must be a matching edge, and thus, the algorithm picks as predecessor of $v_{\ell-i+1}$ the node at the other end of the matched edge that v_{l-i+1} is a part of. Otherwise, the nodes still participating in the algorithm run $\mathsf{MV}(M, \ell - i + 1, f)$ (that is, they run in the graph $G - U_{\ell-i+1}$) The algorithm picks an arbitrary neighbor v of $v_{\ell-i+1}$ satisfying $r^0_{G-U_{\ell-i+1}}(f, v) = \ell - i$ and $\mathbf{I}_M(v, v_{\ell-i+1}) = 0$ as the predecessor of $v_{\ell-i+1}$.

Lemma 10. *Algorithm 2 constructs an augmenting path between f and g with high probability in $O(\ell^2)$ rounds.*

Proof. Let $z_0 = g$ and z_i be the node that satisfies $\mathsf{target} = z_i$ at the end of the i-th iteration for $1 \leq i \leq \ell$. Let H_i be a subgraph induced by $V(G - U_i)$. We prove the statement that $U_{\ell-h}$ is a ($h \mod 2$)-alternating path between z_0 and z_h. As $r^0_G(f, z_\ell) \leq r^0_{H_1}(f, z_\ell) = 0$, $z_\ell = v$ holds, and thus, we obtain U_0 as an augmenting path of length ℓ from f to g by setting $h = \ell$. The proof follows the induction on h. (Basis) Since z_0 chooses the node z_1 that satisfies $\mathbf{I}_M((\mathsf{target}, v_{\ell-1})) = 0$ and $r^0_{H_\ell}(f, v_{\ell-1}) = \ell - 1$ in the first iteration of Algorithm 2, $U_{\ell-1} = \{(z_0, z_1)\}$ is a 1-alternating path between z_0 and z_1. (Inductive Step) As the induction hypothesis, suppose there exists a ($h' \mod 2$)-alternating path between z_0 and $z_{h'}$ at the end of the h'-th iteration. Because $r^{h' \bmod 2}_{H_{\ell-h'+1}}(f, z_{h'}) = \ell - h'$ holds by the definition of $z_{h'}$, there exists an edge $(z_{h'}, v)$ that satisfies $\mathbf{I}_M((z_{h'}, v)) = h' \mod 2$, and $r^{(h'+1) \bmod 2}_{H_{\ell-h'}}(f, v) = \ell - h' - 1$ holds. Therefore, $z_{h'}$ can choose the node $z_{h'+1}$ that satisfies $\mathbf{I}_M((\mathsf{target}, z_{h'+1})) = h' \mod 2$ and $r^0_{H_{\ell-h'}}(f, z_{h'+1}) = \ell - h' - 1$ in the $(h'+1)$-th iteration of Algorithm 2. Hence, $U_{\ell-h'} \circ \{(z_{h'}, z_{h'+1})\}$ is a $((h+1) \mod 2)$-alternating path between z_0 and $z_{h'+1}$ at the end of the $(h'+1)$-th iteration.

We show the running-time analysis of Algorithm 2. The algorithm consists of ℓ iterations. As each iteration is obviously implemented in $O(\ell)$ rounds, the running time of Algorithm 2 is $O(\ell^2)$ rounds. □

Theorem 7 trivially follows from Lemma 10.

3.5 Construction of Augmenting Path in $O(n)$ Rounds

3.5.1 Outline

We first introduce several auxiliary notions and definitions. Given a subgraph $H \subseteq G$ and $\theta \in \{0,1\}$, a node $v \in V_H$ is called *θ-reachable* in H if $r^\theta_H(f,v)$ is finite. In addition, v is called *bireachable* in H if it is both 1-reachable and 0-reachable in H. A node that is neither 1-reachable nor 0-reachable in H is called *unreachable* in H. A node that is θ-reachable for some $\theta \in \{0,1\}$ in H but not bireachable in H is called *strictly θ-reachable* in H. Given two spanning subgraphs H_1 and H_2 of G, we say that a node $v \in V(H_1)$ *preserves the reachability* of H_2 in H_1 if for any $\theta \in \{0,1\}$, the θ-reachability of v in H_2 implies that in H_1. A graph H_1 is said to preserve the reachability of H_2 if any node $v \in V(H_1)$ preserves the reachability of H_2 in H_1, which is denoted by $H_1 \succ H_2$. We define $r_H(f,v) = \min_{\theta\in\{0,1\}} r^\theta_H(f,v)$ and $\gamma_H(v) = \mathrm{argmin}_{\theta\in\{0,1\}} r^\theta_H(f,v)$. Note that $r^0_H(f,v) = r^1_H(f,v)$ does not hold, because $r^0_H(f,v)$ is even and $r^1_H(f,v)$ is odd. When $r^0_H(f,v) = \infty$ and $r^1_H(f,v) = \infty$ hold, $\gamma_H(v)$ is defined as zero. We assume that any node v unreachable from f in G does not join our algorithm. Therefore, without loss of generality, we assume that none of the nodes $v \in V_G$ are unreachable in G without loss of generality. In addition, we assume that any node $v \in V_G$ has information on the values of $r^0_G(f,v)$ and $r^1_G(f,v)$ at the beginning of the algorithm. This assumption is realized by activating $\mathsf{MV}(M,n,f)$ as a preprocessing step.

The key idea of our proof is to construct a *sparse certificate* H, which is a spanning subgraph $H \subseteq G$ of $O(n)$ edges satisfying $H \succ G$. If such a graph is obtained, the trivial centralized approach (i.e., the approach in which f collects the whole topological information of H) yields an $O(n)$-round algorithm for constructing the augmenting path. For constructing sparse certificates, we first introduce a novel tree structure associated with G, M, and f:

Definition 5 (Alternating base tree)**.** *An* alternating base tree *for G, M, and f is a rooted spanning tree T of G satisfying the following conditions:*

- *f is the root of T.*
- *For any $v \in V(G)$, the edge from v to its parent in T is the last edge of the shortest alternating path from f to v in G. Formally, letting $\mathsf{par}_T(v)$ be the parent of $v \in V(G)\setminus\{f\}$ in T, $r^{\gamma_G(v)}_G(f,v) = r^{1-\gamma_G(v)}_G(f,\mathsf{par}_T(v)) + 1$ and $\mathbf{I}_M((v,\mathsf{par}_{T_I}(v))) = 1-\gamma_G(v)$ hold for any $v \in V(G)\setminus\{f\}$.*

It is not difficult to check that such a spanning tree always exists. As a node might have two or more shortest alternating paths, T is not uniquely determined (see Figure 3.1 (1) and (2) for examples). In the following argument, however, we fix an arbitrarily chosen alternating base tree T. It should be emphasized that the alternating base tree does not necessarily contain an alternating path from f to each node v. For example, both alternating base trees in Figure 3.1 have no alternating path from f to v_9.

Fixing T, the subscript T of the notation $\mathsf{par}_T(v)$ is omitted in the following argument. We define $\mathsf{ep}(v)$ as the edge from v to its parent and T_v as the subtree of T rooted by v. We define the *outgoing edges* of T_v as the set of edges whose one of endpoint belongs to $T(v)$ and the other endpoint does not belong to T_v. Any non-tree edge $e = (u,w) \in E(G)\setminus E(T)$ and the unique path from u to w in T form a simple cycle in G, which is denoted by $\mathsf{cyc}(e)$.

The sparse certificate is obtained by incrementally augmenting edges to T. For any $1 \le k \le n$, we define the *level-k edge set* F_k as $F_k = \{(u,v) \mid (u,v) \in E(G)\setminus M \wedge \max(r^0_G(f,u), r^0_G(f,v)) = k\} \cup \{(u,v) \mid (u,v) \in M \wedge \max(r^1_G(f,u), r^1_G(f,v)) = k\}$. We also define $F_{\le k} = \cup_{0\le i\le k} F_k$ and $G_k = T + F_{\le k}$. Moreover, we define $F_0 = \emptyset$ as a sentinel. Let B_k be the set of all the bridges (i.e.,

all the edges forming a cut of size one) in G_k. Note that B_k is a subset of $E(T)$ because T is a spanning tree of G. The following lemma is the key technical ingredient of our construction.

Lemma 11. *Let $F_k^c \subseteq F_k \backslash E(T)$ be an arbitrary subset of non-tree edges in F_k satisfying $B_{k-1} \backslash B_k \subseteq \cup_{e \in F_k^c} E(\mathsf{cyc}(e))$. Then, $(T + \cup_{1 \leq i \leq k} F_i^c) \succ G_k$ holds.*

Lemma 11 naturally yields the following incremental construction of sparse certificates: each node v identifies k such that $\mathsf{ep}(v) \in B_{k-1} \setminus B_k$ holds, and if T_v has an outgoing edge e belonging to F_k, v adds e to F_k^c (if F_k contains two or more outgoing edges, one is chosen arbitrarily). Because $G_k \subseteq G_{k+1}$ holds for any $0 \leq k \leq n-1$, we have $B_{k+1} \subset B_k$, which implies that $B_k \setminus B_{k+1}$ for all k are mutually disjoint. Then, $\sum_{0 \leq i \leq n-1} |B_k \setminus B_{k+1}| = |B_0| = n - 1$ holds. Since at most one edge is augmented for each edge in $B_k \setminus B_{k+1}$, the size $|\cup_{0 \leq i \leq n-1} F_i^c|$ is bounded by $n-1$. Since $\mathsf{cyc}(e)$ obviously covers $\mathsf{ep}(v)$, the constructed edge set F_k^c satisfies the lemma. Consequently, $H = T + \cup_{1 \leq i \leq n} F_i^c \succ G_n$ is satisfied, and thus, H is a sparse certificate.

The idea behind our algorithm is the seminal blossom argument by Edmonds [26]. A walk $W = v_0, e_1, v_1, e_2, \ldots, v_\ell$ is called an *odd (even) alternating cycle* if it satisfies the following condition:

- $\mathbf{I}_M(e_i) + \mathbf{I}_M(e_{i+1}) = 1$ holds for any $1 \leq i \leq \ell - 1$.
- $v_0 = v_\ell$ holds.
- The length of the walk W is odd (even).

If an odd alternating cycle has no consecutive proper subsequence forming an odd alternating cycle, it is called *minimal*.[2] Note that a minimal odd alternating cycle can still have a consecutive subsequence forming an even alternating cycle. The node that is first and last node of odd alternating cycle is called *stem node*. An odd alternating cycle C is said to be *reachable* from an unmatched node x if either the stem node is x or there exists a node v in C admitting an even alternating path from x to v not intersecting C. A node u is called *x-covered* if there exists a minimal odd alternating cycle C reachable from x such that C contains u as a non-stem node. It is known that the vertex v is bireachable in G if and only if v is f-covered in G [26]. Our algorithm adds the edges not in T incrementally with guaranteeing the invariant that v is f-covered in $(T + \cup_{1 \leq i \leq k} F_i^c)$ if and only if v is included in a 2-edge connected component of size at least two in $T + \cup_{1 \leq i \leq k} F_i^c$. The addition of edges from $B_{k-1} \setminus B_k$ in our algorithm can be seen as the process of f-covering the vertex v which is bireachable in G_{k+1} but not in any 2-edge connected component of $(T + \cup_{1 \leq i \leq k} F_i^c)$, by creating new minimal odd alternating cycles reachable from f.

Considering the distributed construction of H, a useful property of Lemma 11 is that one does not have to wait for the computation of F_k^c to start the computation of F_{k+1}^c. As the information on $r_G^\theta(f, v)$ for $\theta \in \{0, 1\}$ is available to v, each node can identify the level of each incident edge. Thus, the construction of F_k^c for all k can be executed in parallel. The details of the distributed construction is explained in Section 3.5.3.

3.5.2 Proof Details

Before proving Lemma 11, we prove an auxiliary lemma.

Lemma 12. *For any $\theta \in \{0, 1\}$ and $v \in V(G) \setminus \{f\}$ such that $r_G^\theta(f, v) \leq k + 1$ holds, $r_{G_{k'}}^\theta(f, v) = r_G^\theta(f, v)$ holds for all $k' \geq k$.*

[2] Due to some technical reason, we allow W to be non-simple, but this modification does not affect the correctness of the original argument by Edmonds.

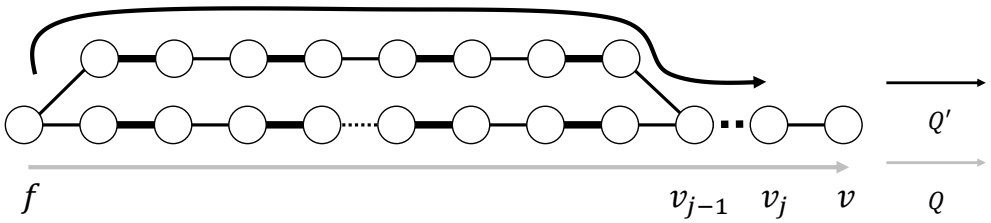

Figure 3.2: Proof of Lemma 12 for (Case 2b). Bold lines are matching edges, and thin lines are unmatched edges. The dotted line is the edge included in G but not in G_k. Note that the edge (v_{j-1}, v_j) is actually included in G_k, but it is drawn with a dotted line for explaining the contradiction.

Proof. The proof is based on induction on k. (Basis) $k = 0$: Let v be any node satisfying $r^\theta_G(f, v) \leq k+1$ for some $\theta \in \{0, 1\}$, and let Q be the θ-shortest path from f to v in G. This path is contained in T because v chooses f as its parent in T. (Inductive Step): As the induction hypothesis, suppose $r^\theta_{G_{k-1}}(f, u) = r^\theta_G(f, u)$ holds (and also $r^\theta_{G_k}(f, u) = r^\theta_G(f, u)$ holds because of $G_{k-1} \subseteq G_k$) for any u and θ satisfying $r^\theta_G(f, u) \leq k$. Consider any node v such that $r^\theta_G(f, v) \leq k+1$ holds. As the case of $r^\theta_G(f, v) < k+1$ is evidently proved by the induction hypothesis, we assume $r^\theta_G(f, v) = k+1$. The proof consists of the following two cases depending on whether the shortest alternating path from f to v is θ-alternating path or not.

(Case 1) $\gamma_G(v) = \theta$: By the definition of alternating base trees, we have $r^{1-\theta}_G(f, \mathsf{par}(v)) = r^\theta_G(f, v) - 1 = k$. In addition, for any $w \in T$, $r^{\gamma_G(w)}_G(f, w) = r^{1-\gamma_G(w)}_G(f, \mathsf{par}(w)) + 1 > r^{\gamma_G(\mathsf{par}(w))}_G(f, \mathsf{par}(w))$ holds. Therefore any node $w \in T_v$ satisfies $r^{\gamma_G(w)}_G(f, w) \geq k+1$. Then, any outgoing non-tree edge of T_v has a level of at least $k+1$. That is, $\mathsf{ep}(v)$ is a bridge in G_k. Since $r^{1-\theta}_G(f, \mathsf{par}(v)) = k$ holds, the induction hypothesis yields $r^{1-\theta}_{G_k}(f, \mathsf{par}(v)) = k$ and thus there exists a $(1-\theta)$-alternating path U from f to $\mathsf{par}(v)$ in G_k. Due to the fact that $\mathsf{ep}(e)$ is a bridge, U does not contain v. Hence the concatenated path $P \circ \mathsf{ep}(e)$ is a θ-alternating path from f to v in G_k of length $k+1$. That is, $r^\theta_{G_k}(f, v) = r^\theta_G(f, v)$ holds.

(Case 2) $\gamma_G(v) = 1-\theta$: Let $Q = v_0, e_1, v_1, e_2, \ldots, e_{k+1}, v_{k+1}$ be the shortest θ-alternating path from f to v in G ($f = v_0$ and $v = v_{k+1}$). To prove the lemma, it suffices to show that any edge in Q has a level of at most k or is an edge in $E(T)$. Suppose for contradiction that a non-tree edge e_j has the level $k' > k$. Without loss of generality, we assume that j is the highest value for which this condition is satisfied. That is, any edge $e_{j'}$ for $j' > j$ has a level at most k or is an edge in T. We define ρ as $\mathbf{I}_M(e_j)$. We further divide Case 2 into the following three subcases depending on whether e_j is the last edge, and otherwise whether it's a matching edge or not.

(Case 2a) $j = k+1$: Since Q is the shortest θ-alternating path of length $k+1$, $\rho = 1-\theta$ holds, and $Q^p_{v_k}$ is a $(1-\theta)$-alternating path from f to v_k of length k. From the condition $\gamma_G(v) = \gamma_G(v_{k+1}) = 1-\theta$ for Case 2, $r^{1-\theta}_G(f, v_k) \leq k$ and $r^{1-\theta}_G(f, v_{k+1}) < r^\theta_G(f, v_{k+1}) = k+1$ hold. That is, the level of $e_j = e_{k+1}$ is at most k, which is a contradiction.

(Case 2b) $j < k+1$ and $\rho = 1$: Since the length of $Q^p_{v_j}$ is j, we have $r^0_G(f, v_j) \leq j \leq k$. From the induction hypothesis, G_{k-1} contains a 0-alternating path Q' from f to v_j. In other words, v_j has a 0-alternating path Q' such that any non-tree edge in $E(Q')$ has a level of at most k. The assumption of $\rho = 1$ implies that Q' must terminate with a matching edge incident to v_j, i.e., the edge e_j. This is a contradiction because we assume that e_j is not contained in G_{k-1}.

(Case 2c) $j < k+1$ and $\rho = 0$: We denote $R = Q^s_{v_j}$ as shorthand. As the length of Q^p_j is $j \leq k$, from the induction hypothesis, we have $r^1_{G_k}(f, v_j) = r^1_G(f, v_j) \leq j$, and thus, G_k contains a 1-alternating

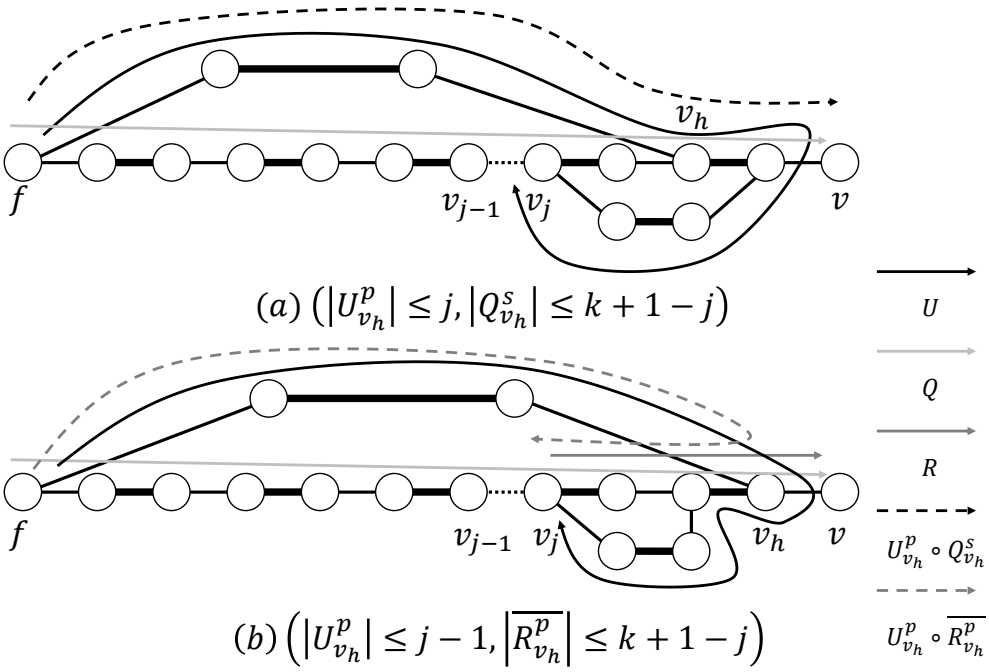

Figure 3.3: Proof of Lemma 12 of (Case 2c). Bold lines are matching edges, and thin lines are unmatched edges. The dotted line is the edge included in G but not in G_k.

path U from f to v_j of shortest length. Let $v_h \in V(R) \cap V(U)$ be the first node in U, which also belongs to R. If e_{h+1} is a matched edge, $U^p_{v_h} \circ Q^s_{v_h}$ is a θ-alternating path in G_k (see Figure 3.3(a)), the length of which is bounded by $|U^p_{v_h} \circ Q^s_{v_h}| \le |U| + (k+1-h) \le j + (k+1-j) \le k+1$. Hence, we obtain $r^\theta_{G_k}(f, v_{k+1}) \le k+1 = r^\theta_G(f, v_{k+1})$, which is a contradiction. If e_{h+1} is an unmatched edge, e_h is a matched edge. Therefore, $U^p_{v_h} \circ \overline{R^p_{v_h}}$ is a 0-alternating path from f to v_j in G_k (see Figure 3.3(b)). Since we consider the case of $\rho = 0$, the edge e_j is an unmatched edge. Therefore, $v_h \neq v_j$ holds, and thus v_h is not the last node of U. This implies $|U^p_{v_h}| \le j-1$. We obtain $|U^p_{v_h} \circ \overline{R^p_{v_h}}| \le j-1+(k+1-h) \le j-1+(k+1-j) \le k$, and thus, $r^0_G(f, v_j) \le r^0_{G_k}(f, v_j) \le k$. Since $Q^s_{v_{j-1}}$ is a 0-alternating path from f to v_{j-1} of length $j-1$, we have $r^0_G(f, v_{j-1}) \le j-1 \le k$. This implies that the level of e_j is at most k, which is a contradiction. □

Now, we present the proof of Lemma 11.

Proof. Let $F^c_{\le k} = \cup_{1 \le i \le k} F^c_i$ and $H_k = T + F^c_{\le k}$. We prove the lemma inductively. For $k = 0$, $H_0 = T \succ G_0 = T$ evidently holds. Thus, it suffices to show $H_k \succ G_k$, assuming $H_{k'} \succ G_{k'}$ for all $0 \le k' < k$. For any $0 \le h \le n$, we define $U_h = \{(v, \theta) \mid v \in V(G) \wedge r^\theta_{G_k}(f, v) = h\}$. If v is θ-reachable in H_k for all $0 \le h \le n$ and $(v, \theta) \in U_h$, we can conclude that $H_k \succ G_k$. The proof of this statement follows the (nested) induction on h. (Basis) As U_0 contains only $(f, 1)$, the statement evidently holds. (Inductive Step) As the induction hypothesis, suppose v is θ-reachable for any $(v, \theta) \in \cup_{0 \le i \le h} U_i$, and consider any pair (v, θ) in U_{h+1}. Then, we consider the following two cases.

(Case 1) $\mathsf{ep}(v)$ is a bridge in G_k: We have $r^{1-\theta}_G(f, \mathsf{par}(v)) = h$ from the definition of alternating base trees. Since the induction hypothesis guarantees that $\mathsf{par}(v)$ preserves the reachability of G_k in H_k, there exists a $(1-\theta)$-alternating path U from f to $\mathsf{par}(v)$ in H_k. In addition, U does not contain $\mathsf{ep}(e)$, because $\mathsf{ep}(e)$ is a bridge in $H_k \subseteq G_k$. From $\mathbf{I}_M(\mathsf{ep}(v)) = 1-\theta$, which directly

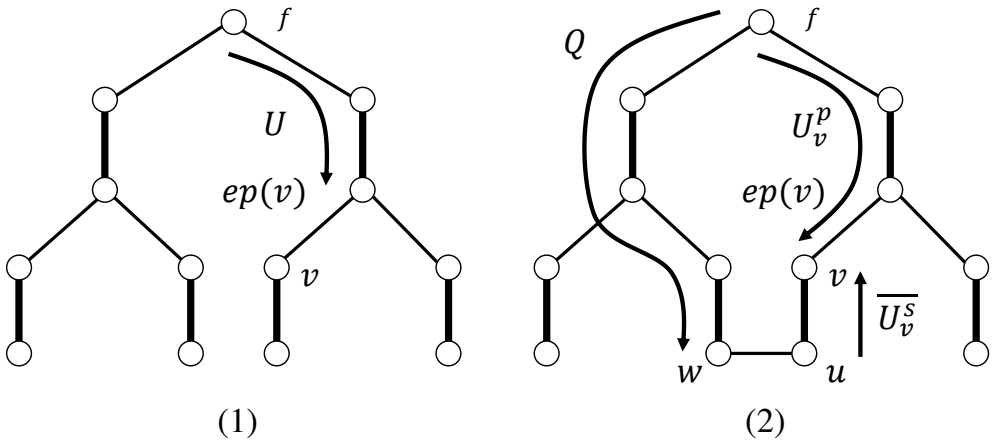

Figure 3.4: Proof of Lemma 11. Bold lines are matching edges, and thin lines are unmatched edges.

Algorithm 3 Construction of the alternating base tree for v_i: $\mathsf{ABT}((G, M))$

Require: The graph induced by the edge set $\bigcup_{i:v_i \in V} E_i$ is an alternating base tree.

1: E_i: initially $\emptyset$.
2: **if** $v_i \neq f$ **then**
3: choose edge (u, v_i) that is incident on the vertex v_i and satisfies $r_G^{\gamma(v_i)}(f, v_i) = r_G^{1-\gamma(v_i)}(f, u) - 1$ and $\mathbf{I}((u, v_i)) = 1 - \gamma(v_i)$ (if multiple edges satisfy these conditions, the node arbitrarily chooses one).
4: $E_i \leftarrow E_i \cup (u, v)$.

follows from the definition of alternating base trees, the concatenated path $U \circ \mathsf{ep}(v)$ becomes a θ-alternating path from f to v in H_k (see Figure 3.4 (1)). Then, v is θ-reachable in H_k.

(Case 2) $\mathsf{ep}(v)$ is not a bridge in G_k: As $G_0 \subseteq G_1 \subseteq \ldots, \subseteq G_k$ holds, there exists $1 \leq j \leq k$ such that $\mathsf{ep}(v) \in B_{j-1} \setminus B_j$ holds. Then, F_j^c contains an outgoing edge e of T_v belonging to F_j. Let $e = (u, w)$ and u be the side contained in T_v. We assume that e is not a matching edge. By symmetry, the case of $e \in M$ is proved similarly. From the definition of F_j, we have $\max\{r_G^1(f, u), r_G^1(f, w)\} = j \leq k$. Lemma 12 implies that both u and w have 1-alternating paths from f in G_{j-1}; from the induction hypothesis $H_{j-1} \succ G_{j-1}$, they have 1-alternating paths from f also in H_{j-1}, which we refer to as U and Q, respectively. Since $\mathsf{ep}(v)$ is a bridge of $G_{j-1} \supseteq H_{j-1}$, the suffix U_v^s is a subgraph of T_v. In addition, Q does not intersect $V(T_v)$, because both f and w are outside T_v. Thus, U_v^s and Q are mutually disjoint, and the concatenated path $Q' = Q \circ (w, u) \circ \overline{U_v^s}$ is simple. It is easy to check that Q' is an alternating path from f to v. As Q, e, and $\overline{U_v^s}$ are all contained in $H_{j-1} + F_j^c = H_j$, U_v^p and Q' are contained in H_j (see Figure 3.4 (2)). The alternating paths U_v^s and Q' have different parities because their last edges are adjacent in U. Hence, we conclude that v is bireachable in H_j.

□

3.5.3 Distributed Implementation

This section explains how to implement the centralized sparse certificate algorithm, presented in Section 3.5.1, in the CONGEST model to obtain the algorithm of Theorem 8. It is relatively straightforward to construct the alternating base tree T. From the preprocessing run of

Algorithm 4 Construction of F_k^c for v_i: $\mathsf{ConstF}(k)$

Require: The edge e_i is an outgoing edge of T_{v_i} if node v_i outputs e_i; otherwise, T_{v_i} does not have an outgoing edge.

for $i = 1; i \leq d; i++$ **do**
  if v_i is a leaf node **then**
    if $I(v_i) \cap F_k \cap E^*(T_v) = \emptyset$ **then**
      $e_{v_i} \leftarrow$ dummy edge e such that $\mathsf{lca}(e) = \infty$.
    else
      $e_{v_i} \leftarrow \min_{e \in I(v_i) \cap F_k \cap E^*(T_{v_i})} e$ w.r.t. $\leq_{\mathsf{lca}}$.
    if $v_i \neq f$ **then**
      send e_{v_i} to its parent.
  else
    if v_i receives the set of edges X from all its children **then**
      $e_{v_i} \leftarrow \min_{e \in X \cup (I(v_i) \cap F_k \cap E^*(T_{v_i}))} e$ w.r.t. $\leq_{\mathsf{lca}}$.
if $\mathsf{lca}(e_{v_i}) \leq d(v_i)$ **then**
  output e_v.
else
  output $\perp$.

$\mathsf{MV}(M, n, f)$, each node v has information on the values of $r_G^1(f, v)$ and $r_G^0(f, v)$; thus, it has information on $\gamma_G(v)$ as well. Then, v chooses an arbitrary neighbor u of v satisfying the second condition of the alternating base tree as its parent (i.e., it chooses (v, u) as an edge of T). Algorithm 3 presents the pseudocode of the alternative base tree construction. This algorithm is a local algorithm, which is implemented in zero round.

The main idea of constructing the edge set $F^c = \cup_{1 \leq i \leq n} F_i^c$ in the distributed manner is implemented by the CONGEST algorithm $\mathsf{ConstF}(k)$, where each node v outputs an outgoing edge of T_v of level k if it exists (or $\perp$ otherwise). Let d be the height of the constructed alternating base tree T. Given a non-tree edge $e = (u, w) \in E(G) \setminus E(T)$, the depth of the lowest common ancestor of u and w is denoted by $\mathsf{lca}(e)$. In addition, we introduce the ordering relation $\leq_{\mathsf{lca}}$ over all non-tree edges as $e_1 \leq_{\mathsf{lca}} e_2$ if and only if $\mathsf{lca}(e_1) \leq \mathsf{lca}(e_2)$. The algorithm ConstF works under the assumption that for any non-tree edge $e = (u, v)$, u and v have information on the value of $\mathsf{lca}(e)$. This assumption is realized by the following $O(d)$-round preprocessing.

1. Each node v computes its depth d_v in T through a downward message propagation from f along T. The root f first sends to its children the value 1. The node v receiving message i decides $d_v = i$ and sends the value $i + 1$ to its children.

2. Each node v broadcasts the pair of its ID and depth (v, d_v) to all the nodes in T_v. First, each node sends the pair to its children. In the following rounds, each node forwards the message from its parents to the children. This task finishes within $O(d)$ rounds.

3. The broadcast information of the previous step allows each node v to identify the path $p_T(v)$ from v to f in T. For all non-tree edges $e = (u, v)$, u and v exchange $p_T(v)$ (taking $O(d)$ rounds) and compute the value of $\mathsf{lca}(e)$.

The pseudocode of Algorithm $\mathsf{ConstF}(k)$ is presented in Algorithm 4. Let $E^*(T_v)$ be the set of non-tree edges e such that at least one endpoint of e belongs to $V(T_v)$. Each node v computes the minimum edge $e_v \in E^*(T_v) \cap F_k$ with respect to $\leq_{\mathsf{lca}}$. This task is implemented through a

standard aggregation over T. Each leaf node v sends the minimum edge e in $F_k \cap E^*(T_v)$ together with $\mathsf{lca}(e)$. If $F_k \cap E^*(T_v) = \emptyset$ holds, the leaf sends a dummy edge e such that $\mathsf{lca}(e) = \infty$ holds . Let X be the set of edges a non-leaf node v received from its children. Then, v chooses e_v as the minimum edge in $X \cup (I(v) \cap F_k \cap E^*(T_v))$ with respect to $\leq_{\mathsf{lca}}$ and sends the chosen edge e_v and $\mathsf{lca}(e_v)$ to $\mathsf{par}(v)$. Finally, v outputs e_v if $\mathsf{lca}(e_v) < d_v$ holds or $\perp$ otherwise. The correctness of $\mathsf{ConstF}(k)$ follows the proposition below.

Proposition 3. *Let e be the minimum edge in $E^*(T_v)$ with respect to $\leq_{\mathsf{lca}}$. Then, e is an outgoing edge of T_v if and only if $\mathsf{lca}(e) < d_v$ holds (thus, $\mathsf{ep}(v)$ is a bridge if $\mathsf{lca}(e) \geq d_v$ holds).*

The edge set F^c is constructed by running $\mathsf{ConstF}(k)$ for all $1 \leq k \leq n$. As this algorithm is implemented by one-shot aggregation over T, one can utilize the standard pipelining technique for completing $\mathsf{ConstF}(k)$ for all $1 \leq k \leq n$, which takes $O(n)$ rounds in total (including the preprocessing step of computing $\mathsf{lca}(e)$). The result of ConstF provides node v with the information of the minimum k, such that $\mathsf{ep}(v) \in B_{k-1} \setminus B_k$, as well as an outgoing edge of T_v in F_k. Following Lemma 12, each node v can decide the edge e that should be added to $F^c = \cup_{1 \leq i \leq n} F_i^c$.

Chapter 4

Uniform Distribution for Pachinko

4.1 Introduction

In this section, we consider the mathematical models of Pachinko in the 50-50 model. The formal definition of the 50-50 model is explained in Section 4.2. The precise statement of our results is as follows:

- For any integer a, we show the construction of the $(1/2^a)$-uniform distribution in the 50-50 model.
- Given any drop-probability distribution A and any partial drop-probability distribution B, we show that it is NP hardness to determine if there exists a pin arrangement that transforms A into B.

4.2 Preliminary

4.2.1 Configuration and Rewriting Rule

The problem was formulated in the 50-50 model using the notion of formal grammar. A Pachinko machine is represented by a triangle grid on a half plane with an infinite horizontal length and an infinite downward vertical length. Each horizontal line contains grid points and is called a *row*. From the top end, each row is assigned a y-coordinate $1, 2, \ldots$. Since the field is a triangle grid, the grid points on an odd row are half-shifted from those on an even row. To fit them into the standard orthogonal coordinate system, these were assigned even x-coordinates to the grid points in even rows, and odd x-coordinates to those in odd rows (see Figure 4.1). Let $\mathbb{N}$ represent the field of natural numbers that contains 0. Any coordinate $(i, j) \in \mathbb{Z} \times \mathbb{N}$ for i and j with a different parity is not a grid point, which is the space for the ball to drop down to the lower rows. These coordinates are called *passages*. In addition, a ball is a point of radius zero. Initially, the ball is dropped from the horizontal center from the top. Hence, the probability that the ball passes through $(0, 0)$ is one. The *drop probability* of the i-th row is denoted as the probability distribution that represents the probability that the ball passes each column. The probability that the ball passes through (i, j) is called the drop probability of (i, j). In addition, $dp(i, j)$ is denoted as the drop probability of (i, j). A pin can be placed at a grid point (i, j) which satisfies $i = j \mod 2$. In the 50-50 model, if the dropping ball hits a pin at point (i, j) (i.e., passes through $(i, j - 1)$), it moves to $(i - 1, j + 1)$ with a probability of $1/2$ and $(i + 1, j + 1)$ with a probability of $1/2$. Therefore, $dp(i - 1, j + 1) = dp(i - 1, j) + dp(i, j)/2$, $dp(i, j + 1) = 0$, and $dp(i + 1, j + 1) = dp(i, j)$

hold. If no pin is spiked at (i, j), the drop probability of $(i, j+1)$ is equal to (i, j). Therefore, $dp(i, j+1) = dp(i, j)$ holds.

A *pin arrangement* P is a set of grid points where the pins are spiked. It generates the drop probability distribution for all of the coordinates in the i-th row for all $i \geq 1$. The distribution of the i-th row is called the i-th *configuration* of P (or simply a configuration). Formally, a configuration is a finite odd-length sequence of rational values whose sum is equal to one. In addition, the center of the sequence corresponds to the drop probability at x-coordinate zero and two infinite sequences of zeros spanning x-coordinates $\pm\infty$ are cut off. It is easy to see that the drop probability for any coordinate (i, j) in a configuration has a form of $y/2^x$, where x is greater than or equal to 0 and y is a non-negative odd integer. The value x is referred to as the *granularity* of the configuration. Given a set $\mathcal{P}$ of pin arrangements, its granularity can be defined as the minimum granularity of all of the configurations generated by the pin arrangements in $\mathcal{P}$. By only considering a set of pin arrangements with the minimum granularity g, any configuration can be treated as a sequence of non-negative integer values by multiplying each probability by 2^g.

The change of configurations (i.e., the change of the corresponding probability distribution) by placing a pin at a grid point is expressed as an application of rewriting the rules to the configurations. Even though two or more can be placed in the same row, the pin placement is equivalently translated into the placement in a number of rows where each row contains at most one pin. Thus, without loss of generality, it is assumed that each row contains at most one pin. By the same reason, it is not necessary to consider the parity of the coordinate (i, j). By inserting a pin coordinate (i, j) and the parities of i and j are different, then a pin coordinate $(i, j+1)$ can be placed instead. Each configuration can be regarded as a *word* over the symbol set $[0, 2^g]$. If a pin is placed at a grid point with the coordinate (i, j), the probability mass of the coordinate $(i, j-1)$ is evenly split into $(i-1, j+1)$ and $(i+1, j+1)$. This can be expressed by rewriting the rule as follows.

Definition 6.

$$uvw \to \left[u + \frac{v}{2}\right] 0 \left[w + \frac{v}{2}\right] \qquad \text{(Rule R1)}.$$

The brackets [ ] represent the single symbol that corresponds to the arithmetic value inside. The symbols u or w may be an implicit zero value that are omitted in the representation of the configurations. An example of rewriting is illustrated in Figure 4.1.

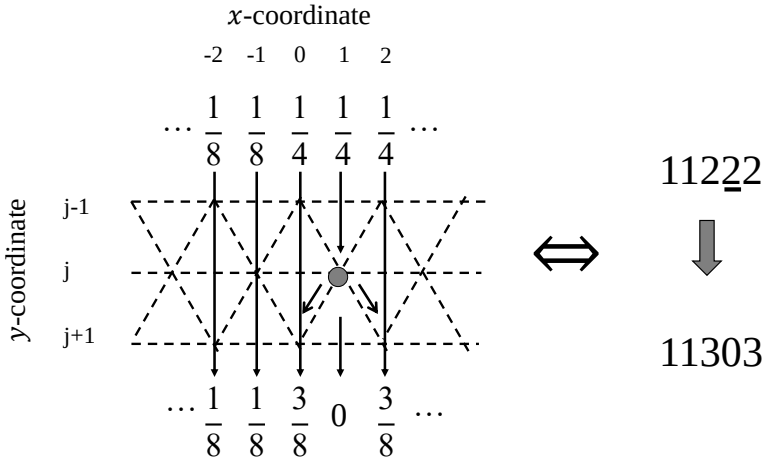

Figure 4.1: An example of the configurations ($g = 3$ and j is an odd integer) and rewriting.

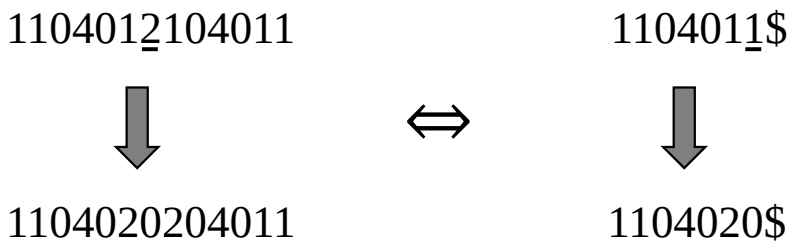

Figure 4.2: An example of rewriting the symmetric configurations.

4.2.2 Symmetric Configuration

Throughout this study, only the symmetric configurations are considered. In other words, the configurations are mirror-symmetric with respect to the vertical center line. To express the symmetric configurations, the right side from the center line is not necessary. This is denoted by $W[v/2]\$$, which is a symmetric configuration $W[v]W^R$. W is a string over the alphabet $\{0, 1, 2, \ldots, [2^g]\}$, v is a symbol that is evaluated as an even number, W^R is the inverted string of W, and \$ is a special symbol that represents the right side from the center line (the *boundary*). Except for the center, any rewriting is applied symmetrically. In other words, when transforming symmetric configurations, putting a pin at (i, j) implies putting another pin at $(-i, j + 2)$. The exception to this case is rewriting at the center, which is handled by a special rule below (note that the drop probability at the center is expressed by its half).

Definition 7.

$$vw\$ \to [v + w]\, 0\$ \qquad \text{(Rule R2)}.$$

The symbol \$ corresponds to the right side from the center line. Figure 4.2 is an example of how the symmetric configurations are rewritten by rule R2. By generating the configuration $W'\$$ from $W\$$ through a finite number of applications of rewriting rules, then $W'\$$ is transformed from $W\$$, and this is written as $W\$ \rightsquigarrow W'\$$. This notion of transformability was also extended into the substring cases. Let $UVW\$$ and $UV'W\$$ represent two words such that V and V' have the same length. If $UVW\$ \rightsquigarrow UV'W\$$ holds for the strings U and W, then V' is transformed from V, and this is written as $V \rightsquigarrow V'$.

4.2.3 Formulation of the Problem

This section formalizes the problem of generating uniform distributions. The goal of this problem is to generate the probability distribution of $1/2^a$, $1/2^a$, $\cdots$,$1/2^a$, 0, $1/2^a$, $1/2^a$, $\cdots$, and $1/2^a$. This is called the $(1/2^a)$-*uniform distribution*. In this construction, the minimum granularity $1/2^{a+1}$ of the drop probability suffices to generate the $(1/2^a)$-uniform distribution; thus, the problem is reduced to the transformability of a single number $[2^a]\$$ to $2^{(2^{a-1})}0\$$, where $2^{(2^{a-1})}$ represents the string of 2^{a-1} repetition of symbol 2. Note that $[2^b]$ and 2^b are not the same word. The former one is a word that consists of a single symbol $[2^b]$, where 2^b represents the arithmetic function of 2 to the power of b. The problem that this study solves is a recursive version of this transformability. The main result is stated by the following theorem.

Theorem 9. *Let* $k = 2^{a-2}$, $4^k 0\$ \rightsquigarrow 2^{2k}0\$$ *holds for any* $a \geq 5$.

In [7], it has been proven that the $(1/2^a)$-uniform distribution can be generated for $a \leq 4$. By applying theorem 9 iteratively, the $(1/2^a)$-uniform distribution can be generated for any $a \geq 1$.

4.3 Generating Uniform Distribution

Section 3 is devoted to the proof of the theorem 9. The proof consists of the following three parts:

1. $4^k 0\$ \rightsquigarrow (440)^{\frac{k}{2}}\$$.
2. $(440)^{\frac{k}{2}}\$ \rightsquigarrow 42^{k-3}02^{k-1}4\$$.
3. $42^{k-3}02^{k+1}4\$ \rightsquigarrow 2^{2k}0\$$.

The combination of these transformations results in the theorem 9. The following subsections looks at the details of each part.

4.3.1 Part 1: From $4^k0\$$ to $(440)^{\frac{k}{2}}\$$

First, the preliminary lemma is explained.

Lemma 13. *Let u, v and w represent any symbols. For any $j \geq 3$, the following transformations are possible:*

$$uv^j w \rightsquigarrow [u+v]0v^{j-2}0[w+v]. \tag{4.1}$$

$$uv^j 0\$ \rightsquigarrow [u+v]0v^{j-1}0\$. \tag{4.2}$$

Proof. The transformation (4.1) was first considered. The proof is based on the induction on j. (Basis) In the case of $j = 3$, the following transformation was obtained (each underline represents the position of rewriting. If there are multiple underlines, these were rewritten from left to right.):

$$\begin{aligned} u\underline{v}v\underline{v}w &\rightsquigarrow \left[u+\frac{v}{2}\right] 0\underline{[2v]}0 \left[w+\frac{v}{2}\right] \\ &\rightsquigarrow \left[u+\frac{v}{2}\right] \underline{v}0\underline{v} \left[w+\frac{v}{2}\right] \\ &\rightsquigarrow [u+v]0v0[w+v]. \end{aligned}$$

(Inductive step) Suppose for the induction hypothesis that the transformation (4.1) is possible for $j = h \geq 3$. The case of $j = h+1$ is obtained as follows:

$$\begin{aligned} uv^{h+1}w &= \underline{uv^h}\underline{v}w \\ &\rightsquigarrow [u+v]0y^{h-2}0\underline{[2v]}w \qquad \text{(Induction hypothesis)} \\ &\rightsquigarrow [u+v]0v^{h-2}v0[v+w] \\ &= [u+v]0v^{h-1}0[v+w]. \end{aligned}$$

Thus, the transformation (4.1) is possible. The proof of the transformation (4.2) follows by rewriting the process below:

$$\begin{aligned} u\underline{v^j}0\$ &\rightsquigarrow [u+v]0v^{j-2}0\underline{v}\$ \qquad \text{(Transformation (4.1))} \\ &\rightsquigarrow [u+v]0v^{j-2}v0\$ \\ &= [u+v]0v^{j-1}0\$. \end{aligned}$$

The lemma holds. □

For simplicity of the argument, for any word A, an appropriate number of zeros was put to the left side of A so that the number of zeros in A becomes $k/2+1$. The i-th *run* of A ($1 \leq i \leq k/2$) is the substring between i-th zero and $(i+1)$-th zero (indexed from the left end of A). The length of the i-th run in A is denoted by $l_A(i)$. Now the notion of the normal forms (NFs) can be defined, which is the class of configurations that was treated in the proof of part 1.

Definition 8. *A word A is in* normal form *(NF) with respect to k if and only if every run in A consists of only the symbol 4, the number of runs (4) is at most $k/2$, and the symbol neighboring the boundary is* 0.

The *run-length vector* $vec(A)$ of word A is the $k/2$-dimensional vector whose j-th element corresponds to $l_A(j)$. Let $vol_A(h) = \sum_{j \in [1,h]} l_A(j)$. Then, strongly-normal form (SNF) is defined as follows.

Definition 9. *A NF word A (with respect to k) is in* strongly-normal form *(with respect to k) if and only if it satisfies $vol_A(h) \leq 2h$ for any $h \in [1, k/2]$.*

Note that $4^k0\$$ and $(440)^{k/2}\$$ are both in SNF. For any two SNF words A_1 and w_2, this study defines $c(A_1, A_2)$ to be the minimum index such that $l_{A_1}(c(A_1, A_2)) \neq l_{A_2}(c(A_1, A_2))$ holds, and define $\mathcal{N}_k$ as the set for all SNF words with respect to k. Then, the order $\preceq$ is defined over $\mathcal{N}_k$ as the lexicographic order of the corresponding run-length vectors. In other words, the following is defined:

$$\begin{cases} A_1 = A_2 & \text{if } A_1 = A_2, \\ A_1 \prec A_2 & \text{if } A_1 \neq A_2 \text{ and } l_{A_1}(c(A_1, A_2)) < l_{A_2}(c(A_1, A_2)). \end{cases}$$

The lexicographic order is known as the total order. For any SNF word A, let $t(A)$ represent the position of the leftmost run with a length more than two, i.e., $t(A) = \min_{j \in [1,k/2], l_A(j) \geq 3} j$. If no run has a length more than two, this is defined as $t(A) = k/2 + 1$. The rewriting process of $4^k0\$ \rightsquigarrow (440)^{k/2}\$$ is to iterate the application of lemma 13 (1) with the prefix u of A before the $t(A)$-th run and the suffix w of A after the $t(A)$-th run if $t(A) < k/2$, or (2) with the prefix u of A before the $t(A)$-th run if $t(A) = k/2$ to the $t(A)$-th run, until the transformation reaches the word A' with $t(A') = k/2 + 1$. This process correctly creates $(440)^{k/2}\$$.

Lemma 14. *Let A be any SNF word, and A' is the word obtained by the process above. Then, A' is also in SNF and $A \prec A'$.*

Proof. It is easy to check that any run for A' consists of only 4s and the symbol 0 is the neighbor of \$ in A'. By the definition of SNF, for any SNF word A, $l_A(1) \leq 2$ holds; thus, $t(A) > 1$ holds. This implies that the applications of lemma 13 to the $t(A)$-th run in A does not change the number of runs (of 4s). Consequently A' is in NF. The application of lemma 13 for the i-th run of a word A increases $l_A(i-1)$ and $l_A(i+1)$ by one, and decreases $l_A(i)$ by two. As a result, the value of $vol_{A'}$ is as follows:

$$vol_{A'}(h) = \begin{cases} vol_A(h) + 1 & \text{if } h = t(A) - 1, \\ vol_A(h) - 1 & \text{if } h = t(A), \\ vol_A(h) & \text{otherwise.} \end{cases}$$

To show that A' is in SNF, it suffices to prove $vol_{A'}(t(A)-1) \leq 2(t(A)-1)$. Because $l_A(t(A)) \geq 3$, this results in $vol_A(t(A)-1)+3 \leq vol_A(t(A)) \leq 2t(A)$; thus, $vol_A(t(A)-1) \leq 2(t(A)-1)-1$ holds. Since the length of the $t(A)$-th run increases at most by one after the application of lemma 13, the

following is obtained: $vol_{A'}(t(A)-1) \leq vol_A(t(A)-1)+1 \leq 2(t(A)-1)$. Therefore, A' is in SNF. By the definition, $c(A,A') = t(A)-1$ holds; thus, this leads to: $l_A(c(A,A')) > l_A(c(A,A'))$, that is $A \prec A'$. Therefore, the lemma is proven. □

Lemma 15. *The word* $(440)^{k/2}\$$ *is the maximum element with respect to* $\preceq$.

Proof. Let $A = (440)^{k/2}\$$. Suppose for the contradiction that a SNF word A' satisfies $A \neq A'$ and $A \prec A'$. Then, $vol_w(c(A,A')) < vol_{A'}(c(A,A'))$ holds. However, since $vol_w(c(A,A')) = 2c(A,A')$ holds, this leads to $vol_{A'}(c(A,A')) > 2c(A,A')$. This contradicts the fact that A' is in SNF. □

The two lemmas above imply that the rewriting process eventually reaches the maximum element of SNF words; hence, the following corollary holds.

Corollary 2. *Let* $k \in \mathbb{N}$ *represent any even positive integer. Then, the following transformation is possible.*

$$0^{k/2}4^k0\$ \rightsquigarrow 0(440)^{\frac{k}{2}}\$.$$

4.3.2 Part 2: From $(440)^{\frac{k}{2}}\$$ to $42^{k-3}02^{k-1}4\$$

This section first introduces a magical string $B_i = 42^i02^{i+2}4\$$, as well as its properties. Before showing the properties of B_i, the preliminary lemmas are presented.

Lemma 16. *Let* u, v, *and* w *represent any symbols, and* j *can represent any positive integer. Then, the following transformations are possible:*

$$u[2v]v^jw \rightsquigarrow [u+v]v^{j-1}0[2v]w. \tag{4.3}$$

$$u[2v]v^jw \rightsquigarrow [u+v]v^j0[w+v]. \tag{4.4}$$

Proof. This study first considered the transformation (4.3). The proof is based on the induction of j. (Basis) In the case of $j = 1$, the following transformation is achieved:

$$u[\underline{2v}]vw \rightsquigarrow [u+v]0[2v]w.$$

(Inductive step) Suppose for the induction hypothesis that the transformation (4.3) is possible for $j = h$. The case of $j = h+1$ is shown as follows:

$$\begin{aligned} u[\underline{2v}]v^{h+1}w &\rightsquigarrow [u+v]\underline{0[2v]v^hw} \\ &\rightsquigarrow [u+v]vv^{h-1}0[2v]w. \qquad \text{(Induction hypothesis)} \\ &= [u+v]v^h0[2v]w. \end{aligned}$$

The transformation (4.4) is obtained by the following rewriting process:

$$\begin{aligned} \underline{u[2v]v^jw} &\rightsquigarrow [u+v]v^{j-1}0\underline{[2v]}w \qquad \text{(Transformation (4.3))} \\ &\rightsquigarrow [u+v]v^j0[w+v]. \end{aligned}$$

Therefore, the lemma holds. □

Corollary 3 is the symmetric version of the lemma 16. Note that by having $W \rightsquigarrow W'$, then $W^R \rightsquigarrow W'^R$ can also be transformed, where W^R and W'^R are inverted strings of W and W'^R, respectively.

Corollary 3. *Let u, v, and w represent any symbols, and j can be any positive integer. Then, the following transformations are possible:*

$$uv^j[2v]w \rightsquigarrow u[2v]0v^{j-1}[w+v]. \tag{4.5}$$

$$uv^j[2v]w \rightsquigarrow [u+v]0v^j[w+v]. \tag{4.6}$$

Lemma 17. *Let j be a positive integer that is greater than or equal to four. Then, $02^j4\$ \rightsquigarrow 2^202^{j-2}4\$$ holds.*

Proof. In addition, $02^j4\$$ can be rewritten as follows:

$$\begin{aligned}
02^j4\$ &= \underline{02^{j-1}2}4\$ \\
&\rightsquigarrow 202^{j-3}0\underline{44}\$ && \text{(Lemma 13 (4.1), } u=0,\ v=2,\ w=2) \\
&\rightsquigarrow 202^{j-3}0\underline{80}\$ \\
&\rightsquigarrow 2\underline{02^{j-3}40}4\$ \\
&\rightsquigarrow 2202^{j-3}24\$ && \text{(Corollary 3 (4.6), } u=0,\ v=2,\ w=0) \\
&= 2^202^{j-2}4\$.
\end{aligned}$$

Therefore, the lemma holds. □

The goal of part 2 is to obtain B_{k-3} from $(440)^{\frac{k}{2}}$. This study introduces two important properties of $B_i = 42^i02^{i+2}4\$$, which is the primary reason why B_i is "magical."

Lemma 18. *Let w be any symbol, and i is any positive integer. Then, the following transformations are possible:*

$$w4B_i \rightsquigarrow [w+4]0B_i. \tag{4.7}$$

$$0440B_i \rightsquigarrow B_{i+2}. \tag{4.8}$$

Proof. This study first considered the transformation (4.7). In the case of $i=1$, this leads to:

$$\begin{aligned}
w4B_1 &= w4\underline{4}202^34\$ \\
&\rightsquigarrow w60\underline{4}02^34\$ \\
&\rightsquigarrow w62\underline{02^4}4\$ \\
&\rightsquigarrow w\underline{62^30}2^24\$ && \text{(Lemma 17)} \\
&\rightsquigarrow w\underline{8}0202^34\$ && \text{(Lemma 13 (4.1), } u=6,\ v=2,\ w=0) \\
&\rightsquigarrow [w+4]04202^34\$ \\
&= [w+4]0B_1.
\end{aligned}$$

In the case of $i \geq 2$, the following transformation is obtained:

$$\begin{aligned}
w4B_i &= w4\underline{4}2^i02^{i+2}4\$ \\
&\rightsquigarrow w6\underline{042^{i-1}0}2^{i+2}4\$ \\
&\rightsquigarrow w62^i\underline{02^{i+3}}4\$ && \text{(Lemma 16 (4.4), } u=0,\ v=2,\ w=0) \\
&\rightsquigarrow w\underline{62^{i+2}0}2^{i+1}4\$ && \text{(Lemma 17)} \\
&\rightsquigarrow w\underline{8}02^i02^{i+2}4\$ && \text{(Lemma 13 (4.1), } u=6,\ v=2,\ w=0) \\
&\rightsquigarrow [w+4]042^i02^{i+2}4\$ \\
&= [w+4]0B_i.
\end{aligned}$$

The transformation (4.8) is obtained by the following rewriting process:

$$\begin{aligned}
0440B_i &= 044\underline{042^i}02^{i+2}4\$ \\
&\rightsquigarrow 0442^{i+1}02^{i+3}4\$ && \text{(Lemma 16 (4.4), } u=0,\ v=2,\ w=0) \\
&= \underline{04B_{i+1}} \\
&\rightsquigarrow 40B_{i+1} && \text{(Transformation (4.7), } w=0) \\
&= 4\underline{042^{i+1}}02^{i+3}4\$ \\
&\rightsquigarrow 42^{i+2}02^{i+4}4\$ && \text{(Lemma 16 (4.4), } u=0,\ v=2,\ w=0) \\
&= B_{i+2}.
\end{aligned}$$

As a result, the lemma holds. □

Why are these properties so important? Our intuitive understanding of lemma 18 (4.7) is that it can treat B_i as a \$. Precisely, the lemma is presented below.

Lemma 19. *Let u and v be any symbols, and i and j can be a positive integer that is greater than or equal to three. Then, the following transformations are possible.*

$$04^j0B_i \rightsquigarrow 404^{j-1}0B_i.$$

Proof. The lemma is proven by the following transformation:

$$\begin{aligned}
0\underline{4^j}0B_i &\rightsquigarrow 404^{j-2}0\underline{4B_i} && \text{(Lemma 13 (4.1), } u=0,\ v=4,\ w=0) \\
&\rightsquigarrow 404^{j-2}40B && \text{(Lemma 18 (4.7), } w=0) \\
&= 404^{j-1}0B.
\end{aligned}$$

Therefore, the lemma holds. □

In Part 1, it was proven that any SNF word W with respect to i can be transformed into $(440)^{i/2}\$$ by only using lemma 13 (4.1) ($u=0$, $v=4$, $w=0$) and lemma 13 (4.2) ($u=0$, $v=4$). The two behaviors of lemma 13 (4.2) ($u=0$, $v=4$) and lemma 19 are the same. This fact yields the corollary below.

Corollary 4. *Let $j \in \mathbb{N}$ be an even positive integer, W represents a SNF word with respect to j, and W' is the string obtained from W by deleting \$. For any positive integer i, $W'B_i \rightsquigarrow (440)^{j/2}B_i$ holds.*

Combining this corollary with lemma 18 (4.8), it can be demonstrated that B_i can recursively "absorb" substring 440 to make itself mature. The following lemma corresponds to the base case of this absorption process.

Lemma 20.

$$(440)^4\$ \rightsquigarrow 44440B_1.$$

Proof. The lemma is proven by the following transformation:

$$
\begin{aligned}
(440)^4\$ &= (440)^2 44\underline{0}4\underline{4}0\$ \\
&\rightsquigarrow (440)^2 602602\$ \\
&\rightsquigarrow (440)^2 60\underline{2}6\underline{2}0\$ \\
&\rightsquigarrow (440)^2 610\underline{8}01\$ \\
&\rightsquigarrow (440)^2 61\underline{4}0\underline{4}1\$ \\
&\rightsquigarrow (440)^2 630\underline{4}03\$ \\
&\rightsquigarrow (440)^2 63\underline{2}0\underline{2}3\$ \\
&\rightsquigarrow (440)^2 6\underline{4}0204\$ \\
&\rightsquigarrow (440)^2 802204\$ \\
&= 440440\underline{8}02204\$ \\
&\rightsquigarrow 4404440\underline{422}04\$ \\
&\rightsquigarrow 44044\underline{422}24\$ && \text{(Lemma 16 (4.4), } u=0,\ v=2,\ w=0) \\
&\rightsquigarrow 4404\underline{4}6020224\$ && \text{(Lemma 13 (4.1), } u=4,\ v=2,\ w=0) \\
&\rightsquigarrow 44060\underline{8}020224\$ \\
&\rightsquigarrow 4406\underline{4}0\underline{4}20224\$ \\
&\rightsquigarrow 440\underline{8}040\underline{4}0224\$ \\
&\rightsquigarrow 4404044202224\$ \\
&= 44404B_1 \\
&\rightsquigarrow 44440B_1. && \text{(Lemma 18 (4.7), } w=0)
\end{aligned}
$$

□

The following two lemmas are the main body of part 2, which proposes the rewriting process by absorbing substring 440.

Lemma 21. *Let i and j be any positive integer. Then, $0^i(440)^iB_j \rightsquigarrow B_{j+2i}$ holds.*

Proof. The proof is based on the induction on i. (Basis) In the case of $i = 1$, the following transformation is obtained:

$$0440B_j \rightsquigarrow B_{j+2}, \qquad \text{(Lemma 18 (4.8))}$$

and in the case of $i = 2$, the following transformation is achieved:

$$
\begin{aligned}
0^2(440)^2B_j &= 0^244\underline{0440}B_j \\
&\rightsquigarrow 0^244\underline{B_{j+2}} && \text{(Lemma 18 (4.8))} \\
&\rightsquigarrow 0^2\underline{8}0B_{j+2} && \text{(Lemma 18 (4.7), } w=4) \\
&\rightsquigarrow 0404\underline{B_{j+2}} \\
&\rightsquigarrow 0440B_{j+2} && \text{(Lemma 18 (4.7), } w=0) \\
&\rightsquigarrow B_{j+4}. && \text{(Lemma 18 (4.8))}
\end{aligned}
$$

(Inductive step) Suppose for the induction hypothesis that lemma 21 holds for $i = h$ ($h \geq 2$). The case of $i = h + 1$ is proven by:

$$\begin{aligned}
0^{h+1}(440)^{h+1}B_j &= 0^{h+1}(440)^{h-2}440440\underline{440B_j} && \text{(Because } h \geq 2\text{)} \\
&\rightsquigarrow 0^{h+1}(440)^{h-2}44044\underline{B_{j+2}} && \text{(Lemma 18 (4.8))} \\
&\rightsquigarrow 0^{h+1}(440)^{h-2}4408\underline{0}B_{j+2} && \text{(Lemma 18 (4.7), } w = 4\text{)} \\
&\rightsquigarrow 0^{h+1}(440)^{h-2}44404\underline{B_{j+2}} && \\
&\rightsquigarrow \underline{0^{h+1}(440)^{h-2}44440B_{j+2}} && \text{(Lemma 18 (4.7), } w = 0\text{)} \\
&\rightsquigarrow \underline{0^{h}(440)^{h}B_{j+2}} && \text{(Corollary 4,} \\
& && W' = 0(440)^{h-2}44440, \\
& && j = 2h\text{)} \\
&\rightsquigarrow B_{j+2(h+1)}. && \text{(Induction hypothesis)}
\end{aligned}$$

Therefore, the lemma holds. □

Lemma 22. *For any even integer $k \geq 8$, $(440)^{\frac{k}{2}}\$ \rightsquigarrow B_{k-3}$ holds.*

Proof.

$$\begin{aligned}
(440)^{\frac{k}{2}}\$ &= (440)^{\frac{k}{2}-4}\underline{(440)^4}\$ \\
&\rightsquigarrow (440)^{\frac{k}{2}-4}44440B_1. && \text{(Lemma 20)}
\end{aligned}$$

The word $(440)^{\frac{k}{2}-4}44440$ is in SNF (with respect to $k-2$). Thus, it can be rewritten as follows:

$$\begin{aligned}
\underline{(440)^{\frac{k}{2}-4}44440B_1} &\rightsquigarrow (440)^{\frac{k}{2}-2}B_1 && \text{(Corollary 4, } W' = (440)^{\frac{k}{2}-4}44440, \\
& && j = k-4\text{)} \\
&\rightsquigarrow B_{k-3}. && \text{(Lemma 21)}
\end{aligned}$$

Therefore, the lemma holds. □

4.3.3 Part 3: From $42^{k-3}02^{k-1}4\$$ to $2^{2k}0\$$

Finally, it is proven that $42^{k-3}02^{k-1}4\$$ can be transformed into $2^{2k}0\$$. This section explains the four preliminary lemmas that were used.

Lemma 23. *Let ℓ and j be any positive integers, and i is a positive integer greater than or equal to two. Then, the following transformation is possible:*

$$02^{\ell}02^{2i}02^{j} \rightsquigarrow 02^{\ell+i-1}02^{2}02^{j+i-1}.$$

Proof. The proof is based on the induction on i. (Basis) In the case of $i = 2$, this results in the following transformation:

$$02^{\ell}\underline{02^{4}0}2^{j} \rightsquigarrow 02^{\ell+1}02^{2}02^{j+1}. \qquad \text{(Lemma 13 (4.1), } u = 0,\ v = 2,\ w = 0\text{)}$$

(Inductive step) Suppose for the induction hypothesis that lemma 23 holds for $i = h \geq 2$. The case of $i = h + 1$ is proven by:

$$\begin{aligned} 02^{\ell}\underline{02^{2(h+1)}0}2^{j} &\rightsquigarrow \underline{02^{\ell+1}02^{2h}02^{j+1}} && \text{(Lemma 13 (4.1), } u = 0, v = 2, w = 0) \\ &\rightsquigarrow 02^{\ell+h}02^{2}02^{j+h}. && \text{(Induction hypothesis)} \end{aligned}$$

As a result, the lemma holds. □

Lemma 24. *For any $i \geq 5$, $02202^{i}4\$ \rightsquigarrow 2^{i-4}02202^{4}4\$$ holds.*

Proof. The proof is based on the induction for i. (Basis) In the case of $i = 5$, the following transformation is obtained:

$$\begin{aligned} 022\underline{02^{5}4}\$ &\rightsquigarrow \underline{022202^{3}4}\$ && \text{(Lemma 17)} \\ &\rightsquigarrow 202202^{4}4\$. && \text{(Lemma 13 (4.1), } u = 0, v = 2, w = 0) \end{aligned}$$

(Inductive step) Suppose for the induction hypothesis that lemma 24 holds for $i = h \geq 5$. The case of $i = h + 1$ is proven by:

$$\begin{aligned} 022\underline{02^{h+1}4}\$ &\rightsquigarrow \underline{022202^{h-1}4}\$ && \text{(Lemma 17)} \\ &\rightsquigarrow 2\underline{02202^{h}4}\$ && \text{(Lemma 13 (4.1), } u = 0, v = 2, w = 0) \\ &\rightsquigarrow 2^{h-3}02202^{4}4\$. && \text{(Induction hypothesis)} \end{aligned}$$

Therefore, the lemma holds. □

Lemma 25. *Let w be any symbol, and i can be any positive integer. Then, $w2^{i}\$ \rightsquigarrow [w+2]2^{i-1}0\$$ holds.*

Proof. The proof is based on the induction on i. (Basis) In the case of $i = 1$, the following transformation is achieved:

$$w\underline{2}\$ \rightsquigarrow [w+2]0\$.$$

(Inductive step) Suppose for the induction hypothesis that lemma 25 holds for $i = h \geq 1$. For the case of $i = h + 1$, the following is obtained:

$$\begin{aligned} w2^{h+1}\$ &= w\underline{22^{h}}\$ \\ &\rightsquigarrow \underline{w42^{h-1}0}\$ && \text{(Induction hypothesis)} \\ &\rightsquigarrow [w+2]2^{h-1}0\underline{2}\$ && \text{(Lemma 16 (4.4), } u = w, v = 2, w = 0) \\ &\rightsquigarrow [w+2]2^{h}0\$. \end{aligned}$$

The case of $i = h + 1$ is proven; therefore, the lemma holds. □

Lemma 26.

$$0220222224\$ \rightsquigarrow 222222220\$.$$

Proof. The lemma is proven by the following transformation:

$$
\begin{aligned}
022\underline{0222}24\$ &\rightsquigarrow 022202044\$ && \text{(Lemma 13 (4.1), } u=0,\ v=2,\ w=2)\\
&\rightsquigarrow 0222020\underline{8}0\$\\
&\rightsquigarrow 022202\underline{4}04\$\\
&\rightsquigarrow 022202\underline{4}40\$\\
&\rightsquigarrow 022202\underline{6}02\$\\
&\rightsquigarrow 02220\underline{2}\underline{6}20\$\\
&\rightsquigarrow 022210\underline{8}01\$\\
&\rightsquigarrow 02221\underline{4}0\underline{4}1\$\\
&\rightsquigarrow 022230\underline{4}03\$\\
&\rightsquigarrow 02223\underline{2}0\underline{2}3\$\\
&\rightsquigarrow \underline{022240}204\$\\
&\rightsquigarrow 202222204\$ && \text{(Corollary 3 (4.6), } u=0,\ v=2,\ w=0)\\
&\rightsquigarrow 2\underline{02222240}\$\\
&\rightsquigarrow 22\underline{0222222}\$ && \text{(Corollary 3 (4.6), } u=0,\ v=2,\ w=0)\\
&\rightsquigarrow 222222220\$. && \text{(Lemma 25, } u=0)
\end{aligned}
$$

□

The combination of these four lemmas deduces the main lemma of part 3.

Lemma 27. *Let k be any even positive integer that is greater than or equal to eight. The following transformation is possible:*

$$0042^{k-3}02^{k-1}4\$ \rightsquigarrow 2^{2k}0\$.$$

Proof. This leads to the following transformation:

$$
\begin{aligned}
0\underline{042^{k-3}}02^{k-1}4\$ &\rightsquigarrow \underline{02^{k-2}}02^{k}4\$ && \text{(Lemma 16 (4.4), } u=0,\ v=2,\ w=0)\\
&\rightsquigarrow \underline{202^{k-4}02}2^{k}4\$ && \text{(Lemma 13 (4.1), } u=0,\ v=2,\ w=0)\\
&\rightsquigarrow 2^{\frac{k}{2}-2}\underline{02^{2}02^{\frac{3k}{2}-2}4}\$ && \text{(Lemma 23, } \ell=1,\ i=k-4,\ j=1)\\
&\rightsquigarrow 2^{\frac{k}{2}-2}2^{\frac{3k}{2}-6}02^{2}02^{4}4\$ && \text{(Lemma 24)}\\
&= 2^{2k-8}\underline{02^{2}02^{4}4}\$\\
&\rightsquigarrow 2^{2k-8}222222220\$ && \text{(Lemma 26)}\\
&= 2^{2k}0\$.
\end{aligned}
$$

□

4.4 Analysis of the Number of Pins for Generating a Uniform Distribution

This section provides the asymptotic bound for the number of pins used in the construction. The numbers of pins used in the transformations shown in the presented lemmas and corollaries are

summarized in Table 4.1. Most of the analyses in the table are easy to check. There are some exceptions that includes the corollary 2, lemma 21, and lemma 22. These analyses are presented below.

Table 4.1: The number of pins used in the transformations.

	Before	After	#pins
Lemma 13	$uv^j w$	$[u+v]0v^{j-2}0[w+v]$	$O(j)$
Lemma 13	$uv^j 0\$$	$[u+v]0v^{j-1}0\$$	$O(j)$
Corollary 2	$0^{k/2}4^k 0\$$	$0(440)^{\frac{k}{2}}\$$	$O(k^3)$
Lemma 16	$u[2v]v^j w$	$[u+v]v^{j-1}0[2v]w$	$O(j)$
Lemma 16	$u[2v]v^j w$	$[u+v]v^j 0[w+v]$	$O(j)$
Corollary 3	$uv^j[2v]w$	$u[2v]0v^{j-1}[w+v]$	$O(j)$
Corollary 3	$uv^j[2v]w$	$[u+v]0v^j[w+v]$	$O(j)$
Lemma 17	$02^j 4\$$	$2^2 02^{j-2}4\$$	$O(j)$
Lemma 18	$w4B_i$	$[w+4]0B_i$	$O(i)$
Lemma 18	$0440B_i$	B_{i+2}	$O(i)$
Lemma 20	$(440)^4\$$	$44440B_1$	$O(1)$
Lemma 21	$0^i(440)^i B_j$	B_{j+2i}	$O(i^3+ij^2)$
Lemma 22	$(440)^{\frac{k}{2}}\$$	B_{k-3}	$O(k^3)$
Lemma 23	$02^\ell 02^{2i}02^j$	$02^{\ell+i-1}02^2 02^{j+i-1}$	$O(i^2)$
Lemma 24	$02202^i 4\$$	$2^{i-4}02202^4 4\$$	$O(i^2)$
Lemma 25	$w2^i\$$	$[w+2]2^{i-1}0\$$	$O(i^2)$
Lemma 26	$022022224\$$	$222222220\$$	$O(1)$
Lemma 27	$0042^{k-3}02^{k-1}4\$$	$2^{2k}0\$$	$O(k^2)$

In the transformation of the corollary 2, the lemma 13 is repeatedly applied until it becomes inapplicable for any 4s run. Recall that lemma 13 is always applied to the leftmost run of 4 whose length is more than or equal to 3. In this repetition, the length of the run is 4 where the lemma 13 applied is three, except for the case where the application is for the $(k/2)$-th run.

Lemma 28. *The number of pins used in the transformation of the corollary 2 is $O(k^3)$.*

Proof. To transform $4^k 0\$$ to $(440)^{\frac{k}{2}}\$$, this study only used lemma 13. Let c_i represent the number of applications of lemma 13 for the i-th run, A be $0^{k/2}4^k 0\$$ and A' be $0(440)^{\frac{k}{2}}\$$. This results in the following equation:

$$l_A(i) = \begin{cases} 0 & \text{if } i \neq \frac{k}{2}, \\ k & \text{if } i = \frac{k}{2}, \end{cases}$$
$$l_{A'}(i) = 2.$$

By applying the lemma 13 to the i-th run, the lengths of the $(i+1)$-th and $(i-1)$-th runs respectively increase by one, and the length of the i-th run decreases by two. Then, the following equation is obtained:

$$l_{A'}(i) = \begin{cases} l_A(i) - 2c_i + c_{i+1} & \text{if } i = 1, \\ l_A(i) + c_{i-1} - 2c_i + c_{i+1} & \text{if } 1 \leq i \leq \frac{k}{2} - 1, \\ l_A(i) + c_{i-1} - c_i & \text{if } i = \frac{k}{2}. \end{cases}$$

This study obtained $c_1 = 0$ and $c_i = c_{i-1} + 2(i-1)$ for $2 \leq i \leq k/2$. In the rewriting process, lemma 13 was applied to the run with a length of three except for the $(k/2)$-th run. Hence, the $O(1)$ pins are consumed per one application. This study applied lemma 13 to the $(k/2)$-th run where the length is at most equal to k. Thus, the number of pins used in that application is $O(k)$. The total number of pins used in the corollary 2 is $O\left(\sum_{i=1}^{\frac{k}{2}-1} c_i + kc_{k/2}\right) = O(k^3)$. □

Lemma 29. *The number of pins used in the transformation of lemma 21 is $O(i^3 + ij^2)$.*

Proof. In the cases of $i = 1$ and $i = 2$, the number of applications of lemma 18 is one and four, respectively. Then, $O(j)$ pins are used. Let $R(i, j)$ represent the number of pins that are used in the transformation from $00(440)^{i-2}44440B_{j+2}$ to $0(440)^iB_{j+2}$. In this transformation, this study only used lemmas 13, 18, and 19. Let c_h be the number of applications of lemmas 13 and 19 to the h-th run, A is $00(440)^{h-2}44440$ and A' is $0(440)^h$. This results in the following equation:

$$l_A(h) = \begin{cases} 2 & \text{if } h \neq \frac{i}{2}, \\ 4 & \text{if } h = \frac{i}{2}, \end{cases}$$
$$l_{A'}(h) = 2.$$

The following equation is obtained:

$$l_A(h) = \begin{cases} l_A(h) - 2c_h + c_{h+1} & \text{if } h = 1, \\ l_A(h) + c_{h-1} - 2c_h + c_{h+1} & \text{if } 1 \leq h \leq \frac{i}{2} - 1, \\ l_A(h) + c_{h-1} - c_h & \text{if } h = \frac{i}{2}. \end{cases}$$

Then, the value of c_i follows $c_1 = 0$ and $c_h = c_{h-1} + 2 = 2(h-1)$. Since lemma 18 is applied c_i times, $R(i, j) = O(\sum_{h=1}^{i+1} c_h + j \cdot c_i) = O(i^2 + ij)$. When $T(i, j)$ is the number of pins used in the transformation from $0^i(440)^iB_j$ to B_{j+2i}, it satisfies the following equality:

$$\begin{aligned} T(i,j) &= O(j^2) + R(i-2, j+2) + T(i-2, j+2) \\ &= O\left(\sum_{h=0}^{\frac{i}{2}}((j+2h)^2 + R(i-2-2h, j+2+2h)\right) \\ &= O\left(\sum_{h=0}^{\frac{i}{2}}((j+2h)^2 + (i-2-2h)^2 + (i-2-2h)(j+2+2h))\right) \\ &= O(i^3 + ij^2). \end{aligned}$$

The lemma is proven. □

The lemma above deduces the following corollary.

Corollary 5. *The number of pins used in the transformation of lemma 22 is $O(k^3)$.*

Putting all the analyses in Table 4.1 together, it can be concluded that the number of pins used in the transformation from $4^k0\$$ to $2^{2k}0\$$ is $O(k^3)$. Thus, the following theorem is obtained.

Theorem 10. *For any $a \geq 5$, there exists a pin arrangement generating the $(1/2^a)$-uniform distribution with $O\left(2^{3a}\right)$ pins.*

4.5 Hardness of Deciding Transformability

4.5.1 Problem Definition

As mentioned in the introduction, a pin arrangement for the 50-50 model can be regarded as a transformer with drop probability distributions. Then, it is a natural question to ask if there exists a pin arrangement that corresponds to the transformation between two given distributions or not. The problem is formally defined as follows.

Problem 1. *Let $A = (p_{-n}, \ldots, p_0, \ldots, p_n)$ and $B = (q_{-m}, \ldots, q_0, \ldots, q_m)$ be two configurations ($p_i, q_i \in \mathbb{Q}$ for any i). Does $A \rightsquigarrow B$ hold?*

Note that the minimum granularity $1/2^g$ is not assumed in the transformability above even though the number of pins used in the transformation does not need to be bounded by a polynomial of n and m.

Unfortunately, the computational complexity of the problem 1 is still unclear even though it is hypothesized to be NP-hard. Instead, a slightly relaxed variant of this problem is introduced, where some specified subset of columns can have arbitrary drop probabilities. For the formal definition, a family of drop probability distributions is defined by the configuration with a special wildcard symbol $*$.

Definition 10. *A partial drop probability distribution $\mathcal{A} = (p_{-n}, \ldots, p_0, \ldots, p_n)$, where $p_i \in \mathbb{Q} \cup \{*\}$ for $i \in [-n, n]$, is the set of drop probability distributions $A = (q_{-n}, \ldots, q_0, \ldots, q_n)$ ($q_i \in \mathbb{Q}$) such that $p_i = q_i$ holds if $p_i \neq *$.*

Given a drop probability distribution A, the expression $A \rightsquigarrow \mathcal{A}$ if the distribution $A' \in \mathcal{A}$ exists such that $A \rightsquigarrow A'$, and if A *can be transformed into* $\mathcal{A}$. A relaxed version of the problem 1 is defined as follows:

Problem 2. *Given a drop probability distribution A and a partial drop probability distribution $\mathcal{A}$, does $A \rightsquigarrow \mathcal{A}$ hold or not?*

In this section, it is proven that problem 2 is NP-hard by the reduction from the subset sum problem.

Definition 11. *(Subset sum) Given a set S of non-negative integer values and a target integer value T, is there a subset $S' \subseteq S$ such that the sum of all elements in S' is equal to T?*

4.5.2 Reduction

Let $I = (S, T)$ be any instance of the subset sum problem, where $S = \{a_1, \ldots, a_N\}$. Let $C = \sum_{i=1}^{N} a_i + T/4$. The drop probability distribution $A(I) = (p_{-\infty}, \ldots, p_0, \ldots, p_\infty)$ and the partial

drop probability distribution $\mathcal{A}(I) = (q_{-\infty}, \ldots, q_0, \ldots, q_\infty)$ are constructed as follows:

$$p_i = \begin{cases} 0 & \text{for any } i \in [-\infty - 1] \text{ and } [2N+1, \infty], \\ 1 - \displaystyle\sum_{i=1}^{N} \left(\frac{1}{2^{i+2}} + \frac{a_i}{2^{3N+6-2i}C} \right) & \text{if } i = 0, \\ \dfrac{1}{2^{2+\frac{i}{2}}} & \text{for any even } i \in [1, 2N], \\ \dfrac{a_{(i+1)/2}}{2^{3N+5-i}C} & \text{for any odd } i \in [1, 2N], \end{cases}$$

$$q_i = \begin{cases} 0 & \text{for any } i \in [-\infty, -1] \text{ and } [2N+2, \infty], \\ \displaystyle\sum_{i=0}^{2N} \frac{p_i}{2^i} + \frac{T}{2^{3N+7}C} & \text{if } i = 0, \\ * & \text{for any } i \in [1, 2N+1]. \end{cases}$$

Figure 4.3 is an example of the reduction, which illustrates the instance $(A(I), \mathcal{A}(I))$ for the problem 2 corresponding to the instance $S = \{3, 7, 1\}$ and $T = 4$.

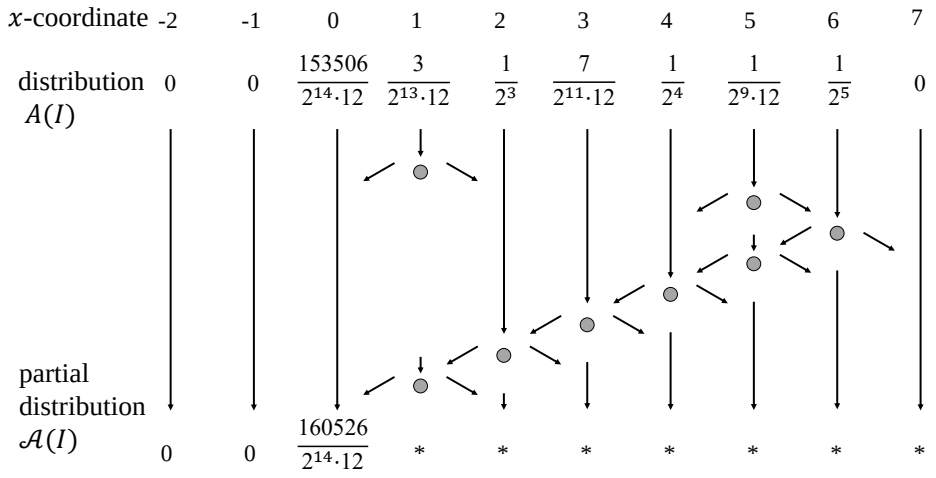

Figure 4.3: An example of the instance $(A(I), \mathcal{A}(I))$ for the problem 2 in the case of $S = \{3, 7, 1\}, T = 4$. The pin arrangement corresponds to the solution $\{3, 1\}$.

4.5.3 Proof

The correctness of the reduction has been proven in this investigation. It is not difficult to check that the reduction can be done in polynomial time (each value p_i can be a high-order value but its length is still bounded by the polynomial of N and $\log C$). Thus, it suffices to show that the proposed reduction preserves the answers between the two problems. First, the sufficiency side is proven.

Lemma 30. *Let I be any yes-instance of the subset sum problem. Then, $A(I) \rightsquigarrow \mathcal{A}(I)$ holds.*

Proof. Let $S' = \{a_{s_1}, \ldots, a_{s_M}\}$ be a solution (certificate) of the instance I. Then, a solution can be constructed $(d_1, \ldots, d_{M+2N})$ for $(A(I), \mathcal{A}(I))$, where d_i corresponds to the x-coordinate for the i-th pin (as assumed in the previous section, each row contains one pin. Thus, for any $i \in [1, M+2N]$, the i-th pin was placed in the y-coordinate i). The value d_i is determined as follows:

$$d_i = \begin{cases} 2s_i - 1 & \text{if } 1 \leq i \leq M \\ 2N + M + 1 - i & \text{if } M+1 \leq i \leq M+2N. \end{cases}$$

The pin arrangement in Figure 4.3 is the construction above for the presented instance. Let $p(j,i)$ be the drop probability at coordinate (j,i). Let $F(i) = \sum_{j=-\infty}^{2N} p(j,i)/2^j$. It is not difficult to check that $p(0, M+2N+1) = F(M+1)$ holds and $p(-2, M+2N+1)$ and $p(-1, M+2N+1)$ are both zero. Thus, it suffices to show that $F(M+1) = q_0$. For $2 \leq i \leq M$, the following equation holds:

$$\begin{aligned} F(i) &= F(i-1) - \frac{p(d_{i-1}, i-1)}{2^{d_{i-1}}} + \frac{p(d_{i-1}, i-1)}{(2 \cdot 2^{d_{i-1}-1})} + \frac{p(d_{i-1}, i-1)}{(2 \cdot 2^{d_{i-1}+1})} \\ &= F(i-1) + \frac{p(d_{i-1}, i-1)}{2^{d_{i-1}+2}}. \end{aligned}$$

The expression $1 \leq i \leq M$, $p(d_i, i) = p_{2s_i - 1}$ holds. Therefore, the following equation is obtained:

$$\begin{aligned} F(M+1) &= F(1) + \sum_{i=2}^{M+1} \frac{p(d_{i-1}, i-1)}{2^{d_{i-1}+2}} \\ &= F(1) + \sum_{i=1}^{M} \frac{p(d_i, i)}{2^{d_i+2}} \\ &= F(1) + \sum_{i=1}^{M} \frac{p(d_i, i)}{2^{2s_i+1}} \\ &= F(1) + \sum_{i=1}^{M} \frac{p_{2s_i-1}}{2^{2s_i+1}} \\ &= \sum_{i=-\infty}^{2N} \frac{p(i,1)}{2^i} + \frac{p_{2s_i-1}}{2^{2s_i+1}} \\ &= \sum_{i=0}^{2N} \frac{p(i,1)}{2^i} + \frac{p_{2s_i-1}}{2^{2s_i+1}} \\ &= \sum_{i=0}^{2N} \frac{p(i,1)}{2^i} + \frac{a_{s_i}}{2^{3N+5-2s_i+1}C} \cdot \frac{1}{2^{2s_i+1}} \\ &= \sum_{i=0}^{2N} \frac{p_i}{2^i} + \frac{T}{2^{3N+7}C} \\ &= q_0. \end{aligned}$$

As a result, the lemma is proven. □

Next, the necessity side has been proven, i.e., I is a yes-instance if $A(I) \rightsquigarrow \mathcal{A}(I)$ holds. Assume that $(A(I), \mathcal{A}(I))$ has a solution $P = (d_1, \ldots, d_L)$, where d_i is the x-coordinate of the i-th pin. In

the following argument, each pin is referred to as an integer value in $[1, L]$. This study defines $p(i, j)$ so it is similar to the proof of lemma 30. A solution is *minimal* if it does not contain any subsequence, which is also a solution with the same instance. Without loss of generality, it is assumed that P is minimal. Three properties are presented for any minimal solution.

Lemma 31. *If two pins i and j $(i \leq j)$ have the same x-coordinate a, then there exists a pin k such that it satisfies $i \leq k \leq j$ and has x-coordinate $a-1$ or $a+1$.*

Proof. Suppose for the contradiction that any pin $i, i+1, \ldots, j-1$ has x-coordinate neither $a-1$ or $a+1$. Then, $p(j, x) = 0$ holds; thus, a smaller solution P' can be constructed by removing d_j from P. It contradicts the minimality of P. □

Lemma 32. *For any pin i $(i \geq 2)$, there exists a pin j such that $j \geq i$ and $d_j = d_i - 1$ holds.*

Proof. Suppose for the contradiction that a pin i does not satisfy the statement of the lemma. Since no pin put after i has the x-coordinate $(d_i - 1)$, the values $p(d_{i-1}, i), p(d_i, i), p(d_{i+1}, i), \ldots$ do not affect the values of $p(0, L+1)$, $p(-1, L+1)$, and $p(-2, L+1)$. This implies that a smaller solution P' can be constructed by removing the pin corresponding to d_i from P, which contradicts the minimality of P. □

Lemma 33. *For any $i \in [1, L]$, $d_i > 0$ holds.*

Proof. Let's suppose for the contradiction in which j is the largest integer value such that $d_j \leq 0$ holds. Since the solution is minimal, $p(d_j, j)$ is non-zero; thus, the value of $p(d_j - 1, j+1)$ becomes non-zero. Since j is the largest, no pin with x-coordinate smaller than d_j is put after j. It follows that the drop probability $p(d_j - 1, L+1)$ is non-zero. It is a contradiction. □

N' is denoted by the smallest positive x-coordinate such that no pin has that x-coordinate (i.e., the leftmost column where no pin exists). Then, the feasibility of a pin arrangement does not change even if any pin is removed with a x-coordinate larger than N'. This is because the pin does not affect the drop probability of the 0-th column. Since P is minimal, it can be assumed that no pin has a x-coordinate larger than N'. A pin arrangement is *even-restricted* if each even column contains at most one pin, each odd column contains at most two pins, and the pin arrangement is minimal. The following shows that if P is an even-restricted solution, then I is a yes-instance.

Lemma 34. *Assume that P is even-restricted, and let X represents the set of columns with two pins in P. Then, this results in $p(0, L) = \sum_{i=0}^{N'} p_i/2^i + \sum_{i \in X} p_i/2^{i+2}$.*

Proof. Let d_{j_1} and d_{j_2} represent the two pins placed at any odd column x $(j_1 \leq j_2)$, d_i, and d_k represents the pins in the $(x-1)$-th and $(x+1)$-th columns, respectively. By lemmas 31 and 32, $i \geq j_2 \geq k \geq j_1$ holds. This implies that the pin j_1 can be moved into any upper row (Figure 4.4). Thus, it is assumed that P has the same form as the construction into the sufficiency proof of lemma 30. That is, the first $|X|$ pins of P correspond to the upper-side pins in each column $a \in X$. Letting $F(i) = \sum_{j=0}^{N'} p(j, i)/2^j$ by the same argument as the proof of lemma 30, $p(0, L+1) = F(L - |X| + 1)$ holds.

$$\begin{aligned} p(0, L+1) =& F(L - |X| + 1) \\ =& F(1) + \sum_{i \in X} \frac{p_i}{2^{i+2}} \\ =& \sum_{i=0}^{N'} \frac{p_i}{2^i} + \sum_{i \in X} \frac{p_i}{2^{i+2}}. \end{aligned}$$

The lemma is proven. □

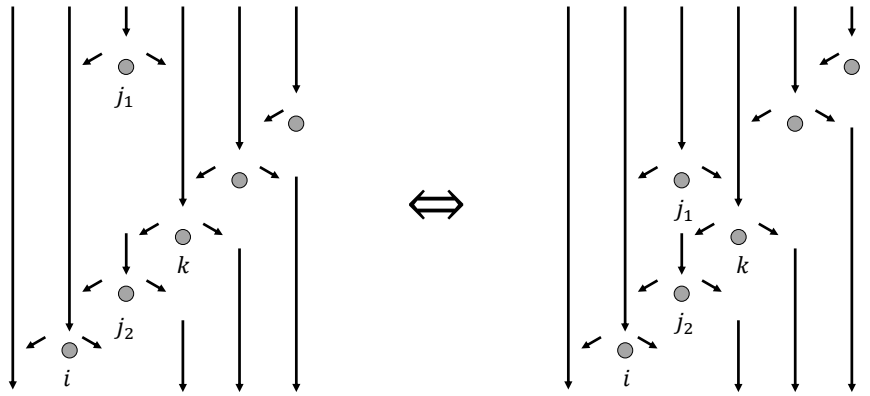

Figure 4.4: An example of the pin movement not affecting the output distribution.

The lemma above implies the following lemma.

Lemma 35. *Let $I = (S, T)$ represent any subset sum instance, and $(A(I), \mathcal{A}(I))$ has an even-restricted solution P. Then, there exists $S' \subseteq S$ such that $\sum_{a \in S'} a = T$.*

Proof. Let X represent the set of columns where two pins are placed. First, it is necessary to show that $N' = 2N + 1$ holds. Since p_i is zero for any $i \geq 2N + 1$ and P is minimal, P does not contain any pin with a x-coordinate larger than $2N + 1$. As a result, $N' \leq 2N + 1$ holds. Let's consider the contradiction that $N' < 2N + 1$. By using the lemma 34, the following inequality is obtained:

$$\begin{aligned} p(0, L+1) &= \sum_{i=0}^{N'} \frac{p_i}{2^i} + \sum_{i \in X} \frac{p_i}{2^{i+2}} \\ &\leq \sum_{i=0}^{2N-1} \frac{p_i}{2^i} + \sum_{i \in X} \frac{p_i}{2^{i+2}} \\ &\leq \sum_{i=0}^{2N-1} \frac{p_i}{2^i} + \sum_{i=0}^{N} \frac{p_{2i-1}}{2^{2i+1}} \\ &< q_0, \end{aligned}$$

which contradicts the fact that P is a solution. Next, the construction of S' is demonstrated. By applying the lemma 34 and $p(0, L + 1) = q_0$, the following equality holds:

$$\begin{aligned} \frac{T}{2^{3N+7}C} &= q_0 - \sum_{i=0}^{2N} \frac{p_i}{2^i} \\ &= p(0, L+1) - \sum_{i=0}^{2N} \frac{p_i}{2^i} \\ &= \sum_{i \in X} \frac{p_i}{2^{i+2}} \end{aligned}$$

$$= \sum_{i \in X} \frac{a_{\frac{i+1}{2}}}{2^{3N+7}C}.$$

This implies that $S' = \{a_{(i+1)/2} \mid i \in X\}$ is a solution of I; hence, the lemma is proven. □

Finally, let's prove that P is even-restricted.

Lemma 36. *For any $k \in [1, L]$ and $l \geq 1$, $\sum_{j=0}^{l} p(j, k-1)/2^j \leq \sum_{j=0}^{l} p(j,k)/2^j$ holds.*

Proof. Let $\Delta(l,k) = \sum_{j=0}^{l} \frac{p(j,k)}{2^j} - \sum_{j=0}^{l} \frac{p(j,k-1)}{2^j}$ for short. This results in:

$$\Delta(l,k) = \begin{cases} -\frac{p(d_k,k-1)}{2^{d_k}} + \frac{p(d_k,k-1)}{2 \cdot 2^{d_k-1}} + \frac{p(d_k,k-1)}{2 \cdot 2^{d_k+1}} & \text{if } d_k \leq l-1 \\ -\frac{p(d_k,k-1)}{2^{d_k}} + \frac{p(d_k,k-1)}{2 \cdot 2^{d_k-1}} & \text{if } d_k = l \\ \frac{p(d_k,k-1)}{2 \cdot 2^{d_k+1}} & \text{if } d_k = l+1 \\ 0 & \text{if } d_k > l+1. \end{cases}$$

In any case, it can be concluded that $\Delta(l,k) \geq 0$ holds. Therefore, the lemma is proven. □

Lemma 37. *Let $P[i] = (d_i, \cdots, d_L)$. For any $d_y \in P[i]$, $p(0,L) \geq \sum_{j=0}^{d_y} p(j, i-1)/2^j$ holds.*

Proof. Let's show by the induction on x that the lemma holds for any $x \in [1, 2N]$ and d_y ($d_y \in P[i]$) satisfying $d_y \leq x$. (Basis) Let z be the smallest value such that $z \geq i$ and $d_z = x (= 1)$ hold. By applying the lemma 36, this results in $p(0, z-1) \geq p(0, i-1)$. This also results in $p(0,z) = p(0,z-1) + p(1,z-1)/2 \geq p(0,i-1) + p(1,i-1)/2$. Since lemma 33 implies that no pin is put in column zero, $p(0,z) \leq p(0, L+1)$ holds. It follows $p(0,L+1) \geq p(0,i-1) + p(1,i-1)/2$. (Inductive Step) Suppose for the induction hypothesis that the lemma holds for some x. Then, let z be the smallest value such that $z \geq i$ and $d_z = x+1$ holds. Then, this results in:

$$\begin{aligned} \sum_{j=0}^{x} \frac{p(j,z)}{2^j} &= \sum_{j=0}^{x} \frac{p(j,z-1)}{2^j} + \frac{p(x+1,z-1)}{2 \cdot 2^x} \\ &\geq \sum_{j=0}^{x} \frac{p(j,i-1)}{2^j} + \frac{p(x+1,i-1)}{2^{x+1}} \\ &= \sum_{j=0}^{x+1} \frac{p(j,i-1)}{2^j}, \end{aligned}$$

where the second inequality is obtained by the repeated application of lemma 36. By using the lemma 32, y' exists such that $d_{y'} \in P[z+1]$ and $d_{y'} = x$ holds. Combined with the induction hypothesis, this results in: $p(0,L+1) \geq \sum_{j=0}^{d_{y'}} p(j,z)/2^j = \sum_{j=0}^{x} p(j,z)/2^j \geq \sum_{j=0}^{x+1} p(j,i-1)/2^j$. □

Lemma 38. *Let $U_i = \sum_{j:d_j=i} p(i,j-1)/2$. For any even $i \in [1,2N]$, $U_i < 5p_i/8$ holds.*

Proof. Suppose for the contradiction that $U_i \geq 5p_i/8$ holds for some i. Let $l_1, l_2, \ldots, l_K$ be the values such that $d_{l_j} = i$ holds ($1 \leq j \leq K$). Then, for any l_k, this leads to:

$$\sum_{j=0}^{i-1} \frac{p(j,l_k)}{2^j} = \sum_{j=0}^{i-1} \frac{p(j,l_k-1)}{2^j} + \frac{p(i,l_k-1)}{2 \cdot 2^{i-1}}$$

Let x be the largest value such that $d_x = i$ holds. By using the lemma 36, this results in:

$$\begin{aligned}\sum_{j=0}^{i-1} \frac{p(j,x)}{2^j} &\geq \sum_{j=0}^{i-1} \frac{p_j}{2^j} + \sum_{j=0}^{K} \frac{p(i,l_j-1)}{2^i} \\ &= \sum_{j=0}^{i-1} \frac{p_j}{2^j} + \frac{U_i}{2^{i-1}} \\ &\geq \sum_{j=0}^{i-1} \frac{p_j}{2^j} + \frac{5p_i}{2^{i+2}} \\ &> q_0.\end{aligned}$$

Using the lemma 37, this results in $p(0, L+1) > q_0$, which is a contradiction. □

Lemma 39. *Any solution for P is even-restricted.*

Proof. Let's first demonstrate that any even column has at most one pin. Suppose for the contradiction that K pins $l_1, l_2, \ldots, l_K$ has the same x-coordinate $(2i)$ ($K \geq 2$, $l_j < l_{j+1}$ for any $j \in [1, K]$, and $i \in [1, N]$). Then, $p(2i, l_1 - 1) \geq p_{2i}$ holds. By applying the lemma 31, there exists a pin l' that satisfies $l_1 < l' < l_2$ in either column $(2i-1)$ or $(2i+1)$. Hence, this leads to $p(2i, l_2 - 1) \geq p_{2i}/4$, which results in $U_{2i} \geq 5p_{2i}/8$, which contradicts the lemma 38. Next, let's show that any odd column has at most two pins. Suppose for the contradiction that three or more pins $l_1, \ldots, l_K$ have the same coordinate $(2i-1)$ ($K \geq 2$, $l_j < l_{j+1}$ for any $j \in [1, K]$, and $i \in [1, N]$). Since the solution P is minimal, there must exist three pins j_1, j_2, and j_3 in either column $(2i-2)$ or $(2i)$ such that $l_1 + 1 \leq j_1 \leq l_2 - 1$, $l_2 + 1 \leq j_2 \leq l_3 - 1$ and $j_3 \geq l_3 + 1$ are satisfied. However, it has been demonstrated that two even columns have at most two pins, which is a contradiction. □

Consequently, the following theorem is deduced from lemmas 30, 35, and 39.

Theorem 11. *The problem 2 is NP-hard.*

Chapter 5

Conclusion

In this dissertation, we show the efficient algorithm for local interaction system.

In Chapter 2, we have shown the upper and lower bounds for the round complexity of shortcut construction and MST in doubling dimension-x graphs, diameter-three or four graphs, and bounded clique-width graphs. We presented an $\tilde{O}(D^x)$-round algorithm for any doubling dimension-x graphs. We also presented the algorithms for constructing optimal low-congestion shortcut with quality $\tilde{O}(\kappa_D)$ in $\tilde{O}(\kappa_D)$ rounds for $D = 3$ and 4, which yields the optimal algorithms for MST matching the known lower bounds by Lotker et al. [67]. On the negative side, $O(1)$- clique width does not allow us to have good shortcuts. We conclude this paper by posing three related open problems as follows: (1) Can we have good shortcuts for the k-clique width where $k \leq 5$? (2) While the bounded clique width does not contribute to solving MST efficiently, it seems to provide several edge-disjoint paths (not necessarily short). Can we find any problem that can use the benefit of bounded clique width?

In Chapter 3, we proposed a randomized $O(s_{\max}^{3/2})$-rounds (i.e. $O(n^{3/2})$-rounds) algorithm for computing a maximum matching in the CONGEST model, which is the first one attaining $o(n^2)$-round complexity for general graphs. Our algorithm follows the standard augmenting-path approach, and the technical core lies two fast algorithms of finding augmenting paths respectively running in $O(\ell^2)$ and $O(s_{\max})$ rounds. While we believe that our result is a big step toward the goal of revealing the tight round complexity of the exact maximum matching problem, the gap between the upper and lower bounds are still large. It should be noted that we leave the possibility of much faster augmenting path algorithms. Once an $o(\ell^2)$-round or $o(s_{\max})$-round algorithm of finding an augmenting path is invented, the upper bound automatically improves. This direction is still promising.

In Chapter 4, this investigation proved that $(1/2^a)$-uniform distributions in the 50-50 model can be generated for any $a \geq 0$. This is a positive answer for the open problem posed in [7]. This construction consumes $O(2^{3a})$ pins for generating $(1/2^a)$-uniform distribution. It is still open if more compact generation (with respect to the number of pins) is possible or not. This study did not pay much attention to the number of rows in the generation process. It is crucial to abandon the simplification assumption that each row contains exactly one pin for optimization. Unfortunately, it makes the treatment of the model complicated; however, revealing the minimum number of rows for generation purposes is an interesting open problem. From the viewpoint of the computational complexity, it seems like an important open problem to show the NP-hardness (or polynomial-time decidability) for the Problem 1. In addition, for both problems 1 and 2, it is still open if they belong to NP or not.

Acknowledgment

I have been fortunate to receive assistance from many people. I would especially like to express my gratitude to my supervisor Professor Yoshiaki Katayama for his guidance. I have also received precious advise from Professors of the Graduate School of Information Science and Technology, Osaka University. Among them, I would like to extend the gratitude to Associate Professor Taisuke Izumi for his valuable comments on my work. I would like to thank to Professor Tadashi Wadayama, Professor Nobuhiro Inuzuka, Professor Toshimitsu Masuzawa, Assistant Professor Yonghwan Kim, Associate Professor Yuichi Sudo, Associate Professor Yota Otachi, Mr. Hirotaka Kitagawa, and Mr. Yuya Kawabata for their useful comments on my work. I would like to thank to the staffs and students of Katayama Kim Laboratory, Department of Computer Science, Nagoya Institute of Technology. I would also like to thank to the staffs and students of Algorithm Engineering Laboratory, the Graduate School of Information Science and Technology, Osaka University. I thank to Japan Society for the Promotion of Science (JSPS) for providing financial assistance in the form of KAKENHI to perform my work comfortably. Finally, I thank all of my families for their continuous encouragement and support.

Publications

Journal Paper

1. Naoki Kitamura, Yuya Kawabata, Taisuke Izumi, "Uniform Distribution for Pachinko", Theorical Computer Science, Vol. 839, pages 103-121, 2020.

2. Naoki Kitamura, Hirotaka Kitagawa, Yota Otachi, Taisuke Izumi, "Low-congestion shortcut and graph parameters, Distributed Computing", Vol. 34, No. 5, pages 349-365, 2019.

3. Naoki Kitamura, Taisuke Izumi, "A Subquadratic-Time Distributed Algorithm for Exact Maximum Matching", The Institute of Electronics, Information and Communication Engineers Transactions, Vol. E105-D, No. 3, pages -, 2022 (in press).

Conference Paper

1. Naoki Kitamura, Yuya Kawabata, Taisuke Izumi, "Uniform Distribution on Pachinko", International Conference on Fun with Algorithms (FUN), pages 103-121, Italy, June, 2018.

2. Naoki Kitamura, Hirotaka Kitagawa, Yota Otachi, Taisuke Izumi, "Low-Congestion Shortcut and Graph Parameters", International Symposium on Distributed Computing (DISC), pages 25:1–25:17, Budapest, October, 2019.

3. Naoki Kitamura, Yuya Kawabata, Taisuke Izumi, "Uniform Distribution on Pachinko", Japan Conference on Discrete and Computational Geometry, Graphs, and Games (JCDCG^3), Japan, September, 2016.

References

[1] Pachinko - wikipedia. https://en.wikipedia.org/wiki/Pachinko.

[2] Pachinko japanzone. http://www.japan-zone.com/modern/pachinko.shtml.

[3] Pinball - wikipedia. http://en.wikipedia.org/wiki/Pinball.

[4] Amir Abboud, Keren Censor-Hillel, and Seri Khoury. Near-linear lower bounds for distributed distance computations, even in sparse networks. In *Proceedings of 30nd International Symposium on Distributed Computing (DISC)*, pages 29–42, 2016.

[5] Mohamad Ahmadi and Fabian Kuhn. Distributed maximum matching verification in CONGEST. In *34th International Symposium on Distributed Computing (DISC)*, pages 37:1–37:18, 2020.

[6] Mohamad Ahmadi, Fabian Kuhn, and Rotem Oshman. Distributed approximate maximum matching in the CONGEST model. In *32rd International Symposium on Distributed Computing (DISC)*, pages 6:1–6:17, 2018.

[7] Hugo A Akitaya, Erik D Demaine, Martin L Demaine, Adam Hesterberg, Ferran Hurtado, Jason S Ku, and Jayson Lynch. Pachinko. *Computational Geometry: Theory and Applications*, 2018.

[8] Jin Akiyama and Mari-Jo P. Ruiz. *Pachinko math.In A Day's Adventure in Math Wonderland.* World Scientific, 2008.

[9] Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. Network decomposition and locality in distributed computation. In *Proceedings of 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 364–369, 1989.

[10] Nir Bacrach, Keren Censor-Hillel, Michal Dory, Yuval Efron, Dean Leitersdorf, and Ami Paz. Hardness of distributed optimization. In *2019 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 238–247, 2019.

[11] Reuven Bar-Yehuda, Keren Censor-Hillel, Mohsen Ghaffari, and Gregory Schwartzman. Distributed approximation of maximum independent set and maximum matching. In *36th annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 165–174, 2017.

[12] Ran Ben-Basat, Ken-ichi Kawarabayashi, and Gregory Schwartzman. Parameterized distributed algorithms. In *33rd International Symposium on Distributed Computing (DISC)*, pages 6:1–6:16, 2018.

[13] Aaron Bernstein and Danupon Nanongkai. Distributed exact weighted all-pairs shortest paths in near-linear time. In *Proc. of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, page 334–342, 2019.

[14] Norbert Blum. A new approach to maximum matching in general graphs. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 586–597, 1990.

[15] Paul Bonsma, Marcin Kamiński, and Marcin Wrochna. Reconfiguring independent sets in claw-free graphs. In *Scandinavian Workshop on Algorithm Theory*, pages 86–97, 2014.

[16] Derek G. Corneil and Udi Rotics. On the relationship between clique-width and treewidth. *SIAM Journal on Computing*, pages 825–847, 2005.

[17] David Coudert, Guillaume Ducoffe, and Alexandru Popa. Fully polynomial fpt algorithms for some classes of bounded clique-width graphs. In *Proceedings of the 2018 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2765–2784, 2018.

[18] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, pages 125–150, 2000.

[19] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, pages 77–114, 2000.

[20] Mohit Daga, Monika Henzinger, Danupon Nanongkai, and Thatchaphol Saranurak. Distributed edge connectivity in sublinear time. *arXiv preprint arXiv:1904.04341*, 2019.

[21] Mirela Damian, Saurav Pandit, and Sriram Pemmaraju. Distributed spanner construction in doubling metric spaces. In *Proceedings of the 10th International Conference on Principles of Distributed Systems (OPODIS)*, pages 157–171, 2006.

[22] Erik D Demaine. Playing games with algorithms: Algorithmic combinatorial game theory. In *International Symposium on Mathematical Foundations of Computer Science*, pages 18–33, 2001.

[23] Erik D Demaine, Martin L Demaine, Eli Fox-Epstein, Duc A Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science*, pages 132–142, 2015.

[24] Michal Dory, Yuval Efron, Sagnik Mukhopadhyay, and Danupon Nanongkai. Distributed weighted min-cut in nearly-optimal time. *arXiv preprint arXiv:2004.09129*, 2020.

[25] Jack Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics*, page 125, 1965.

[26] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, pages 449–467, 1965.

[27] Michael Elkin. Distributed approximation: a survey. *ACM SIGACT News*, pages 40–57, 2004.

[28] Michael Elkin. An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. *SIAM Journal on Computing*, pages 433–456, 2006.

[29] Michael Elkin, Arnold Filtser, and Ofer Neiman. Distributed construction of light networks. *arXiv preprint arXiv:1905.02592*, 2019.

[30] Sebastian Forster and Danupon Nanongkai. A faster distributed single-source shortest paths algorithm. In *59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 686–697, 2018.

[31] Eli Fox-Epstein, Duc A Hoang, Yota Otachi, and Ryuhei Uehara. Sliding token on bipartite permutation graphs. In *International Symposium on Algorithms and Computation*, pages 237–247, 2015.

[32] Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. Networks cannot compute their diameter in sublinear time. In *Proc. of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1150–1162, 2012.

[33] Harold N Gabow and Robert E Tarjan. Faster scaling algorithms for general graph matching problems. *Journal of the ACM (JACM)*, pages 815–853, 1991.

[34] Robert G. Gallager, Pierre A. Humblet, and Philip M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, pages 66–77, 1983.

[35] Juan A. Garay, Shay Kutten, and David Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM Journal on Computing*, pages 302–316, 1998.

[36] Mohsen Ghaffari. Near-optimal scheduling of distributed algorithms. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 3–12, 2015.

[37] Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks II: low-congestion shortcuts, mst, and min-cut. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 202–219, 2016.

[38] Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. Near-optimal distributed maximum flow. In *2015 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 81–90, 2015.

[39] Mohsen Ghaffari and Fabian Kuhn. Distributed minimum cut approximation. In *International Symposium on Distributed Computing (DISC)*, pages 1–15, 2013.

[40] Mohsen Ghaffari and Fabian Kuhn. Distributed MST and broadcast with fewer messages, and faster gossiping. In *Proceedings of 32nd International Symposium on Distributed Computing (DISC)*, pages 30:1–30:12, 2018.

[41] Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 784–797, 2017.

[42] Mohsen Ghaffari, Fabian Kuhn, and Hsin-Hao Su. Distributed MST and routing in almost mixing time. In *Proceedings of 31nd International Symposium on Distributed Computing (DISC)*, pages 131–140, 2017.

[43] Mohsen Ghaffari and Jason Li. Improved distributed algorithms for exact shortest paths. In *Proc. of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 431–444, 2018.

[44] Mohsen Ghaffari and Jason Li. New distributed algorithms in almost mixing time via transformations from parallel algorithms. In *Proceedings of 32nd International Symposium on Distributed Computing (DISC)*, pages 31:1–31:16, 2018.

[45] Robert Gmyr and Gopal Pandurangan. Time-message trade-offs in distributed algorithms. In *Proceedings of 32nd International Symposium on Distributed Computing (DISC)*, pages 32:1–32:18, 2018.

[46] Oded Goldreich. Finding the shortest move-sequence in the graph-generalized 15-puzzle is np-hard. In *Studies in complexity and cryptography. Miscellanea on the interplay between randomness and computation*, pages 1–5, 2011.

[47] Bernhard Haeupler, D. Ellis Hershkowitz, and David Wajc. Round- and message-optimal distributed graph algorithms. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 119–128, 2018.

[48] Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. Low-congestion shortcuts without embedding. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 451–460, 2016.

[49] Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. Near-optimal low-congestion shortcuts on bounded parameter graphs. In *Proceedings of 30nd International Symposium on Distributed Computing (DISC)*, pages 158–172, 2016.

[50] Bernhard Haeupler and Jason Li. Faster distributed shortest path approximations via shortcuts. In *32nd International Symposium on Distributed Computing (DISC)*, pages 33:1–33:14, 2018.

[51] Bernhard Haeupler, Jason Li, and Goran Zuzic. Minor excluded network families admit fast distributed algorithms. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 465–474, 2018.

[52] Stephan Holzer and Roger Wattenhofer. Optimal distributed all pairs shortest paths and applications. In *Proc. of the 2012 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 355–364, 2012.

[53] John E Hopcroft and Richard M Karp. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, pages 225–231, 1973.

[54] Takehiro Ito, Erik D Demaine, Nicholas JA Harvey, Christos H Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, pages 1054–1065, 2011.

[55] Tomasz Jurdzinski and Krzysztof Nowicki. MST in $O(1)$ rounds of congested clique. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2620–2632, 2018.

[56] Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical computer science*, pages 9–15, 2012.

[57] Naoki Kitamura, Hirotaka Kitagawa, Yota Otachi, and Taisuke Izumi. Low-congestion shortcut and graph parameters. In *Proccedings of 33rd International Symposium on Distributed Computing (DISC)*, pages 25:1–25:17, 2019.

[58] Shimon Kogan and Merav Parter. Low-congestion shortcuts in constant diameter graphs. *arXiv preprint arXiv:2106.01894*, 2021.

[59] Fabian Kuhn, Thomas Moscibroda andTim Nieberg, and Roger Wattenhofer. Fast deterministic distributed maximal independent set computation on growth-bounded graphs. In *Proccedings of 19th International Symposium on Distributed Computing (DISC)*, pages 273–287, 2005.

[60] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1109557–1109666, 2006.

[61] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *Journal of the ACM (JACM)*, pages 1–44, 2016.

[62] Fabian Kuhn, Thomas Moscibroda, and Rogert Wattenhofer. On the locality of bounded growth. In *Proceedings of the 24th annual ACM symposium on Principles of distributed computing (PODC)*, pages 60–68, 2005.

[63] Shay Kutten and David Peleg. Fast distributed construction of small k-dominating sets and applications. *Journal of Algorithms*, pages 40–66, 1998.

[64] Christoph Lenzen and David Peleg. Efficient distributed source detection with limited bandwidth. In *Proc. of the 2013 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 375–382, 2013.

[65] Jason Li. Distributed treewidth computation. *arXiv*, 2018.

[66] Jason Li and Merav Parter. Planar diameter via metric compression. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 152–163, 2019.

[67] Zvi Lotker, Boaz Patt-Shamir, and David Peleg. Distributed MST for constant diameter graphs. *Distributed Computing*, pages 453–460, 2006.

[68] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. *Journal of the ACM (JACM)*, pages 1–17, 2015.

[69] Amer E Mouawad, Naomi Nishimura, and Venkatesh Raman. Vertex cover reconfiguration and beyond. In *International Symposium on Algorithms and Computation*, pages 452–463, 2014.

[70] Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *Proc. of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 565–573, 2014.

[71] Danupon Nanongkai and Hsin-Hao Su. Almost-tight distributed minimum cut algorithms. In *International Symposium on Distributed Computing (DISC)*, pages 439–453, 2014.

[72] Hiroaki Ookawa and Taisuke Izumi. Filling logarithmic gaps in distributed complexity for global problems. In *Proccedings of 41st International Conference on Current Trends in Theory and Practice of Informatics (SOFSEM)*, pages 377–388, 2015.

[73] Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. A time- and message-optimal distributed algorithm for minimum spanning trees. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 743–756, 2017.

[74] Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. The distributed minimum spanning tree problem. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 2018.

[75] David Peleg and Vitaly Rubinovich. A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction. *SIAM Journal on Computing*, pages 1427–1442, 2000.

[76] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. In *Proceedings of the 43th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 363–372, 2011.

[77] Jan van den Heuvel. The complexity of change. *Surveys in combinatorics*, 409(2013):127–160, 2013.

[78] Vijay V Vazirani. A proof of the MV matching algorithm. *arXiv preprint arXiv:2012.03582*, 2020.

[79] Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, pages 265–279, 1981.

[80] Takeshi Yamada and Ryuhei Uehara. Shortest reconfiguration of sliding tokens on a caterpillar. In *International Workshop on Algorithms and Computation*, pages 236–248, 2016.