Hardness of Approximation Across Different Models of Computation

By

Seri Khoury

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Summer 2024

Hardness of Approximation Across Different Models of Computation

Abstract

Hardness of Approximation Across Different Models of Computation

By

Seri Khoury

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Shafi Goldwasser, Co-chair

Professor Amir Abboud, Co-chair

A graph is a mathematical construct that models pairwise relations between objects through nodes (or vertices) and edges (also known as links) that connect these nodes. Due to the capacity of graphs to encapsulate a wide range of combinatorial structures, there has been a vibrant pursuit of developing fast and efficient graph algorithms to address a broad spectrum of graph-related problems, including covering and packing problems, clustering problems, distance computation, and symmetry breaking.

This thesis explores approximation algorithms for graph problems. The first part is dedicated to distance computation problems, where we present several new hardness of approximation results across different models of computation, including the centralized, distributed, and dynamic models.

The second part focuses on symmetry-breaking problems in Distributed Computing. Here, we devise efficient approximation algorithms for Maximum Independent Set and Maximum Matching in regular graphs, and introduce new hardness of approximation results for Maximum Independent Set.

To my parents Josephine and Edwar, and to my brothers Peter and Shady.

Contents

Acknowledgments

First, I would like to thank my amazing mentors, Shafi and Amir. I thank Shafi for her parental guidance that extended beyond research-related matters. Her incredible support and encouragement made me feel at home and allowed me to pursue my own interests freely. I am grateful for the long conversations we had, whether in her office or during a walk, which provided not only intellectual stimulation but also great humor. Her mentorship has been essential to my academic and personal development, and I am profoundly thankful for the care and wisdom she has shared with me.

I want to thank Amir, whom I met when I was just 15 years old. From my early days as an undergraduate student, Amir has been guiding me, providing invaluable advice and mentorship. I am immensely grateful for the opportunity to work with him and for all the techniques and tools I have learned under his guidance, which have made me a better researcher. The incredible hospitality at the Weizmann Institute was essential to my learning journey. I am also grateful for our great conversations about research and life, and for the (inevitable) soccer breaks while working on research.

I would also like to extend my gratitude to my dissertation committee members, Avishay Tal and Prasad Raghavendra. I thank Avishay for our great conversations over espresso in the mornings and for his unwavering support throughout the years. I thank Prasad for our insightful research discussions and for giving me the opportunity to organize the theory lunch during my first year at Berkeley.

During my studies, I had the opportunity to meet Aaron Schild, who has become one of my main collaborators and close friends. I have learned so much from Aaron. Besides his exceptional mathematical understanding and the great techniques he taught me, Aaron inspired me with his consistent enthusiasm for solving problems. Working with him has been a privilege; not only was I able to learn many new mathematical techniques, but I also discovered great hidden gems in the nature of California through the hiking trips he organized during research weekends.

During the summer of 2023, I had a great opportunity to be an intern at Google Research in Mountain View. I am grateful for the Google team: Manish Purohit, Joshua Wang, Erik Vee, Zoya Svitkina, and Tamas Sarlos. I particularly want to thank my host Manish Purohit; besides being a great collaborator on our Graph Algorithms project, he taught me everything I know about databases and functional dependencies. The journey at Google has opened new research opportunities for me, and I am very thankful for that.

My journey at UC Berkeley opened the door to meeting many amazing friends and fellow graduate students. Including Orr Paradise and Ofer Grossman, with whom I had several deep conversations and spontaneous adventures; James Bartusek and Sam Gunn, with whom I enjoyed several bar-hopping evenings and good laughs; Sidhanth Mohanty and Siqi Liu, with whom I shared several meditation sessions and philosophical life conversations; Thiago Bergamaschi and Abishek Shetty, for the great soccer activities; Nathan Ju, Yunchao Liu, Elizabeth Yang, Chinmay Nirkhe, Nick Spooner, Arun Ganesh, and Tarun Kathuria, for great barbecue events and endless memories; Ishaq Aden-Ali and Omar Alrabiah, for making sure I don't forget speaking Arabic and for their incredible humor; Zoë Bell, Louis Golowich, Lucas Gretta, Christian Ikeokwu, Malvika Joshi, Rachel Lawrence, Xin Lyu, Jarrod Millman, Ansh Nagda, Angelos Pelecanos, Guru-Vamsi Policharla, Bhaskar Roberts, Jonathan Shafer, Francisca Vasconcelos, David Wu, Pasin Manurangsi, Kewen Wu, and Fred Zhang, for their great friendship and endless memories. I hope to stay in touch with all of you.

Last but not least, I would like to thank my parents, Edwar and Josephine, for their consistent support and for making me the person I am today. Words wouldn't suffice to express my gratitude. Beyond the many things I owe to my parents, I want to thank them for bringing my two amazing brothers, Peter and Shady, with whom I've had endless unforgettable memories.

Chapter 1

Introduction

Graph Theory is one of the most studied areas in Computer Science and Mathematics, with its roots tracing back to Leonhard Euler's fascinating work on the Seven Bridges of Königsberg in 1736 [143]. Over time, graph algorithms have become integral to our daily lives, powering applications such as routing algorithms (e.g., Google Maps and Waze) [129, 251], internet networks [59, 200], neural networks [199, 253], bioinformatics [256, 278], and social network analysis [138, 226].

Traditionally, graph algorithms are designed for the classical centralized computation model, where the graph of interest is provided as input to one computer, and the goal is to solve a graph problem on that computer while minimizing resources.

In recent decades, the rise of computer networks has necessitated the exploration of new computational paradigms, such as *Distributed Computing* [213, 235] and *Dynamic Networks* [141, 177, 257]. These models address a wide range of applications for graph problems, particularly those involving communication systems and computation in constantly changing environments. For instance, in the distributed model, the input graph is not available on a centralized computer; rather, the graph itself acts as a communication network of computers that collaboratively solve a graph problem while minimizing communication. In the dynamic model, the input graph changes over time, and the goal is to maintain an up-to-date solution with few updates (i.e., without recomputing the entire solution from scratch).

While each of these models has its unique challenges that guide our problem-solving strategies within that model, to truly understand the inherent challenges in solving graph problems, it is essential to investigate the common challenges that persist *across* different models of computation. This motivates the following question:

What are some of the common challenges in solving graph problems that are shared across different models of computation?

Understanding the challenges in solving graph problems has been a fundamental concern for researchers for decades. The theory of NP-completeness has provided numerous hardness results that demonstrate what cannot be solved efficiently (i.e., in polynomial time) in the centralized model [193]. Additionally, the field of fine-grained complexity has offered insights into why certain problems do not admit linear or quadratic time algorithms, as well as hardness results for the dynamic model [12, 250]. For the distributed model, a successful framework of reductions from communication complexity has been instrumental in yielding several hardness results [237, 252].

Hardness of Approximation: In many applications, obtaining a satisfactory solution doesn't require solving the problem *exactly*; rather, some degree of error is permissible, and an *approximate* solution suffices. Therefore, when demonstrating a challenge in solving a problem by proving a hardness result, we aim for the hardness result to be *robust*, in the sense that the provable challenge persists even when some degree of error is allowed. Achieving robust hardness results, or *hardness of approximation*, is one of the most central and challenging goals in several models of computation (see, for instance, the open questions in [250] and the discussions in [9, 43]).

Our main goal in this thesis is to understand the challenges in obtaining approximate solutions to graph problems, particularly those that persist across different models of computation.

Our Contribution: We explore two categories of graph problems: distance computation and symmetry breaking. In the first part, we present several new hardness of approximation results for distance computation problems across various models of computation, including the centralized, distributed, and dynamic models. These results rely on a connection between short cycles in graphs and distance approximation. At a very high level, the absence of short cycles in a graph makes it difficult to identify shortcuts hidden among much longer detours. This challenge persists across the aforementioned models.

The second part turns to symmetry breaking problems in the distributed model. We demonstrate that, in some cases, approximate versions of these problems enable the development of faster algorithms, thereby reducing their time complexity. Additionally, we present new hardness of approximation results for finding large independent sets in the distributed model.

In Section 1.2, we discuss the first part of the thesis and elaborate on the connection between short cycles and distance approximation. In Section 1.3, we discuss the second part of the thesis and elaborate on our results for symmetry breaking. In the following subsection, we start by describing our models of interest.

1.1 Models and Basic Notations

Centralized Model: In the classical centralized model, the input graph is provided to one computer, and is represented by a nested list. In this representation, there is one master list containing all the nodes of the graph, and each entry in this master list is itself a list containing all the neighbors of that node. This representation results in an input size of $\Theta(n+m)$ words of memory, where n is the number of nodes and m is the number of edges in the graph.[1] This differs from the adjacency matrix representation, where the graph is depicted by an $n \times n$ Boolean matrix that indicates the presence of an edge between each pair of nodes. Observe that the input size in the adjacency matrix representation is $\Theta(n^2)$ bits. Therefore, the nested list representation proves particularly useful and more compact for sparse graphs where $m = o(n^2)$.

Distributed Models: The standard models of distributed graph algorithms are the LOCAL and CONGEST models [213,235]. In both models, there is a synchronized communication network of n nodes that can communicate via communication rounds. In each round, each node can send and receive a message from each of its neighbors and perform some local computation. The goal is for the nodes to collaboratively solve a graph problem while minimizing the number of communication rounds. The difference between the two models is the size of the messages. In the LOCAL model, the size of the messages is unbounded, allowing

[1]Here, we assume that each node's name can be represented by a single word of memory.

for extensive data exchange. Conversely, in the CONGEST model, the size of each message is bounded by $\log n$ bits, which restricts the amount of information that can be transmitted in each round.

Dynamic Networks: The dynamic model [141,177,257] is a variation of the centralized model where the graph can change over time. In each step, either no changes occur, an edge is added to the graph, or an edge is removed from the graph. The goal is to maintain an up-to-date solution to a graph problem while handling each change as efficiently as possible. The time required to handle each change is often referred to as the *update time*.

1.2 Part I: On the Role of Short Cycles in Hardness of Approximation

In this section, we summarize our hardness of approximation results for distance computation problems. We start with a simple example that illustrates the connection between short cycles and distance approximation in Section 1.2.1. Then, in Section 1.2.2, we summarize our results for the centralized and dynamic models, and in Section 1.2.3, we summarize our results for the distributed model.

1.2.1 A Simple Example

A cycle in a graph is a closed loop that starts at a node v, travels along a series of nodes and edges, and returns to the starting node v without traversing any node or edge more than once (except for the starting node v which also closes the cycle).

To illustrate the role of short cycles in computing approximate distances, consider the following simple example. Let G be an unweighted, undirected bipartite graph with sides L and R, that contains a perfect matching between them. That is, the i-th node ℓ_i in L is connected to the i-th node r_i in R by an edge. Furthermore, assume that there are some additional edges in the graph, but we don't have much information about them.

Now, if we're interested in the shortest path to get from ℓ_i to r_i, we can simply use the edge between them, which forms a path of length one. However, suppose the edge between ℓ_i and r_i is blocked and cannot be used—perhaps the graph G represents a network of roads, and the direct road from ℓ_i to r_i is closed. In this

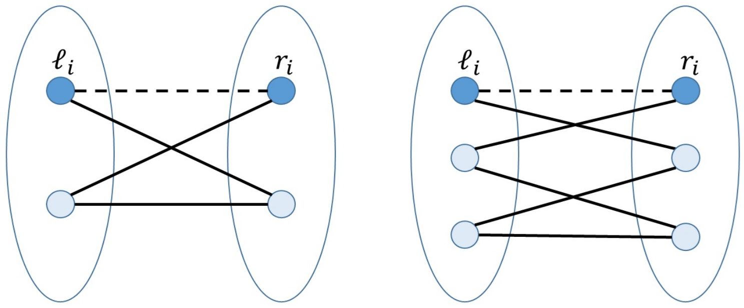

Figure 1.1: On the left, we can replace the edge between ℓ_i and r_i with a zigzag path of length three. However, if the graph is four-cycle free, as shown on the right, then the length of any such zigzag path must be at least five. This increase in path length illustrates the impact of short-cycle freeness on the cost of replacing shortcut edges.

case, to get from ℓ_i to r_i, we could use a *zigzag* path such as $\ell_i \to r' \to \ell' \to r_i$, if the corresponding edges are available. In other words, if our preferred direct path from ℓ_i to r_i is unavailable, we could hope to replace it with a path of length three. Here, we say that the *cost* of replacing the direct path $\ell_i \to r_i$ is three.

Unfortunately, in some cases, that cost can be much higher than three. For instance, if the underlying graph G does not contain four-cycles, then we can't hope to replace an edge with a zigzag of length three, as this would imply the existence of a four-cycle. In this case, the length of the zigzag must be at least five. In general, since G is a bipartite graph, if the length of the shortest cycle (a.k.a. the *girth* of the graph) is at least $k > 2$ for some even k, then the cost of replacing an edge must be at least $k - 1$. For instance, in four-cycle free graphs, the girth is at least six, and the cost is at least five, as discussed. The higher the length of the shortest cycle, the larger the cost becomes. See also Figure 1.1 for illustration.

Interestingly, this connection also works in the other direction. Assume that some edges have been removed from our original graph, and we no longer have access to the updated graph. If we are interested in determining whether the edge $\{\ell_i, r_i\}$ still exists, we can use a black-box algorithm for computing distances to find out. Indeed, we can simply ask our algorithm to compute the distance between ℓ_i and r_i. Moreover, if the girth of the original graph is k, it suffices for the algorithm to tell us whether the distance between ℓ_i and r_i is less than $k - 1$.

If it is, we know that the edge $\{\ell_i, r_i\}$ still exists, as otherwise, the existence of an alternative path of length at most $k-2$ would imply the existence of a cycle of length at most $k-1$ in the original graph, which contradicts the fact that the length of the shortest cycle in the original graph is k.

This is exactly the main idea behind our hardness of approximation results. If we can show that it is hard to determine whether the edges $\{\ell_i, r_i\}$ exist in the updated graph, we can deduce that computing distances is also hard, even when some approximation error is permissible.

Obviously, since in our models of computation we have access to the edges of the graph, determining whether the edges $\{\ell_i, r_i\}$ exist isn't difficult, even if the graph has been updated. To achieve meaningful hardness results, we extend this idea and show that distances can recover not only edges, but also other structures that are hard to identify. Specifically, we show that approximate-distance computation algorithms can be used to recover all the edges that participate in triangles—a problem that is impossible to solve efficiently under popular conjectures (we elaborate on finding triangles in Section 1.2.2). Additionally, we show that a fast distributed algorithm for approximating the maximum distance in the graph (a.k.a. the diameter of the graph) can be used to solve the *Set-Disjointness* problem with little communication, which is known to be impossible (we elaborate on diameter computation in Section 1.2.3). In these reductions, we use the high girth - high cost idea discussed above to show that by using high girth graphs, we can obtain improved hardness of approximation results.

We remark that a conceptually similar idea was used by Feigenbaum et al. [146] to show hardness of diameter approximation in the single-pass streaming model. Applying the high girth - high cost idea to other computational models requires substantially different techniques, which we develop in this thesis.

1.2.2 Finding Triangles via Distance Oracles and Friends

In a joint work with Amir Abboud (thesis co-advisor), Karl Bringmann, and Or Zamir [6], we prove new hardness results for distance computation problems in the centralized and dynamic models. Our results are described in detail in Chapter 2. Here, we provide a brief overview of the main technique, with a particular focus on the Approximate Distance Oracles problem. Let us first discuss the problem and some related work.

Problem Description: In the Approximate Distance Oracles problem, we are given a graph in the centralized model, and the goal is to design a data structure that can preprocess the graph efficiently and then answer distance (i.e., shortest path) queries between pairs of nodes with approximate accuracy. Ideally, the preprocessing step should be completed in linear time relative to the number of nodes and edges, and the response time for each query should be constant or polylogarithmic. The answers provided by the oracle do not need to precisely match the actual distances between the nodes; instead, a constant-multiplicative error is permissible. This allowable error is also referred to as the *stretch* of the requested distances. In this thesis, we explore the possibility of a near-linear preprocessing time for Approximate Distance Oracles, formally given in the following question. Recall than m and n denote the number of edges and nodes in the graph, respectively.

Open Question 1.2.1 (Approximate Distance Oracles)**.** Can we preprocess a graph in $m^{1+o(1)}$ time and answer shortest path queries in $m^{o(1)}$ time with $O(1)$ stretch?

Related Work: This Open Question has been extensively studied (see, for instance, [38, 65, 68, 115, 116, 131, 232, 233, 261, 264, 265] and references within). In 2005, Thorup and Zewick [265] made some progress on resolving this question by providing a data structure with the following properties. For any constant value $k > 0$, the preprocessing step of the data structure runs in $m \cdot n^{1/k}$ time, and the queries are answered in $n^{1/k}$ time with $2k - 1$ stretch. While their solution does not resolve Open Question 1.2.1, it represents a crucial step toward addressing this challenge.

Intriguingly, in a subsequent work by Patrascu, Roditty, and Thorup [233], the authors showed that the upper bound by [265] is optimal for small values of $k < 2$. This implies that for any stretch factor smaller than 3, there is no solution to Approximate Distance Oracles with nearly-linear preprocessing time. However, Open Question 1.2.1 specifically seeks solutions that achieve an arbitrarily constant stretch, and the hardness result presented in [233] does not eliminate the possibility of achieving near-linear preprocessing time for a constant stretch of 3 or greater.

Our Contribution: In this work, we fully resolve Open Question 1.2.1 negatively. Our result is based on the widely recognized 3SUM or APSP conjectures, details of which are differed to Chapter 2.

Theorem 1.2.2. *(Informal) Let $k \geqslant 4$ be an integer. Assuming either the 3SUM or APSP conjectures, there is a positive constant c such that no algorithm can preprocess a graph in $O(m^{1+\frac{1}{ck}})$ time and answer shortest path queries in $O(m^{\frac{1}{ck}})$ time with stretch smaller than k.*

Technical Discussion - The Main Idea Behind the Proof of Theorem 1.2.2: We show that a black-box algorithm for Approximate Distance Oracles can help us recover triangles and solve the *All-Edge Triangle* problem.

In the All-Edge Triangle problem, we are given a tripartite graph G with parts A, B, C and want to detect all edges that are in a triangle $a \in A, b \in B, c \in C$. It is well-known that the 3SUM and APSP conjectures imply that there is no $o(n^2)$ time algorithm for the All-Edge Triangle problem, even in sparse graphs with $m = \Theta(n^{3/2})$ edges [201, 231, 268]. Hence, to prove Theorem 1.2.2, it suffices to reduce All-Edge Triangle to Approximate Distance Oracles.

The standard reduction would do the following. We define a graph G' that is obtained from G by removing the $B \times C$ edges. Then, we query for the distance in G' for any pair $\{b, c\} \in E(G) \cap B \times C$ that used to be an edge in G. If the distance is small, namely 2, we conclude that $\{b, c\}$ is in a triangle in G, because there must be an $a \in A$ that is connected to both b and c; this is the yes-case. Otherwise, if the distance is larger, namely $\geqslant 3$, then we conclude that $\{b, c\}$ is not in any triangle in G because there is no node $a \in A$ that is connected to both b and c; this is the no-case. Given the quadratic lower bound for the All-Edge Triangle problem we conclude that no distance oracle with subquadratic preprocessing time and $m^{o(1)}$ query time can distinguish between distance $= 2$ and $\geqslant 3$.

However, to resolve Open Question 1.2.1 negatively, we must amplify the gap between the distances in the yes-case vs. the no-case. Since the graph G' is bipartite (which is implied by the assumption that G is tripartite) we can readily observe that the distance in the no-case is actually $\geqslant 4$, not just $\geqslant 3$, so the above construction rules out any $(2 - \epsilon)$-approximate answers in the aforementioned time bounds.

Unfortunately from a hardness of approximation perspective, it is rather difficult to argue that the distance in the no-case must be any larger than 4. This is because for any pair $\{b, c\}$ the graph G' is extremely likely to contain a 4-path that makes one *zigzag*, $b \to a \to b' \to a' \to c$, i.e. after the first step from b to

$a \in N(b) \cap A$, it goes back and forth once from $a \in A$ to $b' \in B$ and back to a different node in $a' \in A$, and only then goes to c. (See Figure 1.2.) This path does not imply that $a' \in N(b)$ nor that $a \in N(c)$ and therefore does not correspond to a triangle in G. Indeed, it only corresponds to a 5-*cycle* in G that contains the $\{b, c\}$ edge.

This is exactly where the connection to short cycles comes into play: a short cycle allows a path to make a short detour (a zigzag) and prevents us from achieving a larger gap between the yes- and no-cases. We refer to this challenge as the *Short Cycle Barrier*.

In FOCS 2010, Pătraşcu and Roditty [232] devised an ingenious graph-products technique (conceptually similar to [44]) to push the lower bound to approximation factors beyond 2. Thinking of their construction in the terminology of triangles, their idea is to make G' have $k > 3$ layers by adding $k-2$ layers between B and C that together represent A. In the yes-case where $\{b, c\}$ is in a triangle, the distance is now $k-1$, but the main advantage is that in the no-case they manage to force any path from b to c to make a zigzag in *each of the* $k-1$ *layers*, making the distance $3k-2$. For large enough k this shows that distance oracles with $(3-\epsilon)$-approximations cannot meet the aforementioned time bounds. In the original paper [232], they could only make this approach work for small k and could only prove inapproximability for factors $2\frac{2}{3}-\epsilon$, but in a follow-up paper with Thorup [233] the full potential of this approach was realized, and they established a lower bound for any $(3-\epsilon)$-approximations. Alas, it is clear that 3 is the limit of this approach. (We remark that this is a barrier even in *weighted* graphs.[2])

The natural and more promising approach for circumventing this barrier is to somehow ensure that there are no $\leqslant k$-cycles in the original graph G. Then, any effective zigzag must be *long*, and even the natural two-layered construction would give us a lower bound of $\Omega(k)$. Indeed, the distance for a pair b, c would be $= 2$ if the pair is in a triangle, versus $\geqslant k-1$ otherwise. This would be reminiscent of the use of the girth conjecture in lower bounds for *multiplicative* spanners [31, 238], whereas the aforementioned graph-products technique is reminiscent of lower bounds for *additive* spanners [3, 4, 273].[3] All we have to do is to

[2]The case of *directed* graphs is different however. For some of these problems even deciding if the distance is finite has strong lower bounds. The reason is that the directed edges can prevent zigzags.

[3]The Girth Conjecture and the techniques for additive spanners were already used, of course, for lower bounds against distance oracles as well. However, such lower bounds (and any

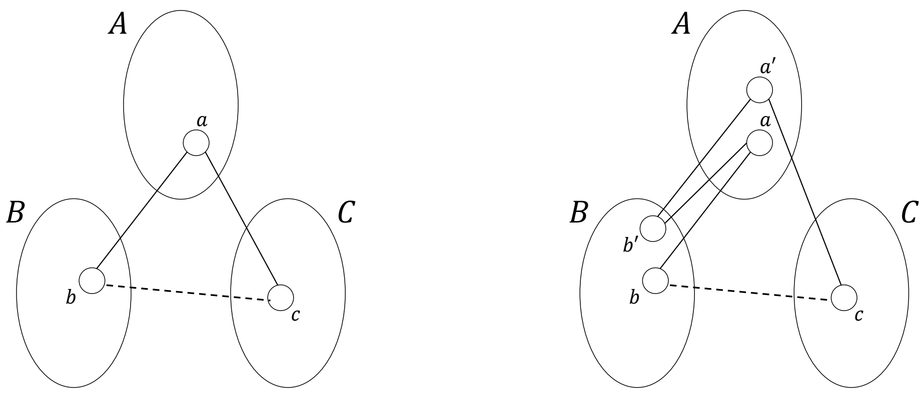

Figure 1.2: If $\{b, c\}$ belongs to a triangle in G, then the distance between b, c in G' is 2. If $\{b, c\}$ does not belong to a triangle in G, then the distance between b, c in G' can be as small as 4. This corresponds to a 5-cycle in G.

prove this gap amplification result for All-Edge Triangle, amplifying the no-case from triangle-free to k'-cycle free, for all $4 \leqslant k' \leqslant k$ (without unintentionally removing a triangle in the yes-case). This is the main idea behind our proof. We show that triangle finding doesn't become easy in short cycle-free graphs, which allows us to prove Theorem 1.2.2.

Our hardness of approximation result for distance oracles extends to dynamic shortest paths, girth approximation, and cycles listing.

Follow-Up Work: In two follow-up works by Abboud, Bringmann, and Fischer [5], and Jin and Xu [189], the authors optimize our short-cycle removal technique and get better lower bounds on the preprocessing time for Approximate Distance Oracles. Specifically, they show that instead of showing that triangle finding doesn't become easy in short-cycle free graphs, they show that 3SUM doesn't become easy even when the input doesn't contain some specific additive structure. Interestingly, without this additive structure, the reduction from 3SUM to triangle finding [231] leaves the graph without many short cycles to begin with. This finding helps the authors of [5,189] improve our bounds and get the complexity of Approximate Distance Oracles closer to the known upper bounds.

information-theoretic arguments) cannot prove lower bounds higher than m; rather, they are interesting for understanding how much dense graphs can be compressed. Thus, the similarity can only be in spirit.

1.2.3 Hardness of Approximation for Distributed Diameter

In this section, we briefly summarize our results for the hardness of approximation of the diameter in the distributed CONGEST model. These results were published in a paper co-authored with Ofer Grossman and Ami Paz [170]. We present the full details of these results in Chapter 3.

Problem Description and Related Work: The diameter D of a graph is the maximum distance between any two nodes in it, which is one of the most fundamental parameters in Graph Theory. In Distributed Computing, the diameter is of utmost importance, as it captures the minimal number of rounds needed for a message to traverse all the nodes in the network. The complexity of computing the exact or approximate value of the diameter has been extensively studied in the distributed setting [7,100,101,155,178,179,181,184,207,236].

In the CONGEST model, the complexity of computing the exact diameter is $\Theta(n/\log n + D)$ rounds [155,184]. On the other hand, there is a folklore algorithm yielding a $1/2$-approximation for the diameter in $O(D)$ rounds: running a BFS (from an arbitrary node) and returning its depth. Moreover, a simple indistiguishability argument shows that no constant-approximation factor is achievable in $o(D)$ rounds.

This raises the following natural question: For values of $\frac{1}{2} < \alpha < 1$, how hard is it to α-approximate the diameter of a graph? Conversely, what is the best approximation factor α that can be achieved in sub-linear time with respect to the number of nodes, and what approximation factors are achievable in sub-polynomial time?

Open Question 1.2.3. For which values of α does there exist a sub-polynomial time α-approximation algorithm for the diameter in the CONGEST model?

Open Question 1.2.4. For which values of α does there exist a truly sub-linear time α-approximation algorithm for the diameter in the CONGEST model?

Our Contribution: We make progress on both these questions. For the first, we show that α must be at most $6/11$. For the second, we show that α must be at most $3/5$. The previous best known upper bound on α, for both cases, was $2/3$ [7]. All the results that are presented in this work, as well as the ones we compare with, are for unweighted and undirected graphs.

Our proofs use the technique of reductions from the Set-Disjointness problem in Communication Complexity to the CONGEST model. At a high level, we show that the Set-Disjointness problem can be embedded into hard instances for diameter computation. These hard instances are short cycle-free, allowing us to leverage a concept similar to the zigzag idea previously described in Sections 1.2.1 and 1.2.2 to obtain improved hardness of approximation.

1.3 Part II: Distributed Symmetry Breaking

In this section, we briefly summarize our results for distributed symmetry breaking. We start with our results for Maximum Independent Set in Section 1.3.1. Then, in Section 1.3.2, we discuss our results for Maximum Matching. Our results for Maximum Independent Set are described in full detail in Chapters 4 and 6, and our results for Maximum Matching are detailed in Chapter 5.

1.3.1 Approximate Maximum Independent Set

One of the most fundamental problems in distributed graph algorithms is the *maximal independent set* problem (MIS), where given an input graph, we need to find a maximal subset of the nodes such that no two nodes in the subset are adjacent. This problem has received a great amount of attention in various distributed models (see for example [21,39,48,61–63,157–159,204,209,213,216,225,228,229,247]). It is considered one of the four classic problems of local distributed algorithms, along with edge coloring, vertex coloring, and maximal matching [60,148,228].

Independent sets are critical in both practical and theoretical aspects of computer science, particularly large independent sets. Applications span various fields including economics [79], computational biology [94,269], coding theory [81,95], and experimental design [46]. In unweighted graphs, a maximum independent set is an independent set of the largest size. In weighted graphs, a maximum-weight independent set (MaxIS) is an independent set with the maximum total weight.

In an unweighted graph, any MIS constitutes a Δ-approximation for MaxIS, where Δ is the maximum degree of a node in the graph. This implies that MIS is no easier than Δ-approximation for MaxIS in unweighted graphs, regardless of the computational model.

This raises a natural question of whether Δ-approximation for MaxIS is easier than MIS. In the classical sequential setting, finding an MIS has the same complexity as finding a Δ-approximation for MaxIS (even in weighted graphs) as both problems admit simple linear-time greedy algorithms.

Our Contribution: In collaboration with Ken-ichi Kawarabayashi, Aaron Schild, and Gregory Schwartzman [197], we show that in the distributed setting, finding a $(1+\epsilon)\Delta$-approximation for MaxIS is exponentially easier than solving MIS. Specifically, we have developed an algorithm that takes $O(\text{poly} \log \log n)$ rounds in the CONGEST model, applicable even to weighted graphs. This running time is impossible for MIS, given the $\Omega(\sqrt{\log n / \log \log n})$ lower bound by Kuhn, Moscibroda, and Wattenhofer [204].

Faster Algorithm for Sparse Graphs: Moreover, we show that if the degree of the graph is less than $n/\log n$, then finding a $(1+\epsilon)\Delta$-approximation for MaxIS requires only $O(1/\epsilon)$ rounds. This results relies on a novel martingales-based analysis of the classical algorithm by Luby [216]. This analytical approach is also integral to the results for Maximum Matching discussed in the subsequent section.

Impossibility Result for Dense Graphs: Finally, we show that the fast running time for sparse graphs cannot be achieved in dense ones. Specifically, we show that any algorithm for finding an $O(\Delta)$-approximation in high-degree graphs must spend at least $\Omega(\log^* n)$ rounds.

Impossibility Results for Constant-Approximation: Finally, in a joint work with Yuval Efron and Ofer Grossman [139], we show linear and quadratic lower bounds for ≈ 2 and $\approx 4/3$-approximation to Maximum Independent Set in the CONGEST model. We elaborate of these results in Chapter 6.

1.3.2 Approximate Matching in Regular Graphs

As discussed in the previous subsection, we show that approximate MaxIS can be solved much more easily than MIS in distributed environments. This raises the question of whether a similar advantage exists for matching problems. A matching in a graph is a set of edges where no two edges share a common endpoint. A *maximal matching* is a matching that cannot be extended by adding more edges, whereas a *maximum matching* is a matching of the largest possible size.

Our Contribution: In a joint work with Manish Purohit, Aaron Schild, and Joshua Wang [198], we found that in regular graphs, it is possible to find a $(1+\epsilon)$-approximation to Maximum Matching within $\text{poly}(1/\epsilon)$ rounds in the CONGEST model. Notably, this runtime is independent of the number of nodes n. Such efficiency is unachievable for Maximal Matching, even in regular graphs, due to a lower bound of $\min\{\log\log n, \Delta\}$ by Balliu et al. [48].

Furthermore, we show that in sufficiently dense regular graphs, where the degree is $\text{poly}(1/\epsilon)$, a $(1+\epsilon)$-approximation to Maximum Matching can be found in only $O(\log(1/\epsilon))$ rounds.

Interestingly, by a simple argument, we show this runtime cannot be generalized to sparse regular graphs. We prove that any algorithm must take $\Omega(1/(\epsilon\Delta))$ rounds in such graphs, where Δ is the degree of the graph. These results underscores the variability in algorithmic efficiency dependent on graph density and structure.

Our main technical contribution is a new lemma, which we refer to as the *Recursive Regularity Lemma*. In this lemma, we show that running Luby's algorithm on the the line graph of a sufficiently dense Δ-regular graph and removing the matched edges together with their incident nodes results in a $(\Delta/2)$-regular graph.

Part I

On The Role of Short Cycles in Hardness of Approximation

Chapter 2

Hardness of Approximation in P via Short Cycle Removal

This chapter is adapted from [6], a joint work with Amir Abboud (thesis co-advisor), Karl Bringmann, and Or Zamir. In this work, we present several new hardness of approximation results in the centralized and dynamic models through reductions from Triangle Finding.

Triangle finding is at the base of many conditional lower bounds in the centralized and dynamic models, and the existence of many 4- or 5-cycles in a worst-case instance has been an obstacle towards resolving major open problems.

We present a new technique for efficiently removing almost all short cycles in a graph without unintentionally removing its triangles. Consequently, triangle finding problems do not become easy even in almost k-cycle-free graphs, for any constant $k \geqslant 4$. This finding allows us to make progress on major open problems in the field of fine-grained complexity, including Approximate Distance Oracles, Dynamic Shortest Paths, Girth Approximation, and Cycle Listing.

2.1 Introduction

One of the most central and challenging goals in fine-grained complexity is to prove *hardness of approximation* results for the many fundamental problems that we already know are hard to compute exactly. With the exception of few results that follow from simple gadget reductions,[1] understanding the time vs. approximation trade-off seems to require specialized fine-grained *gap amplification* techniques. As we know from the quest for *NP-hardness* of approximation that started in the early 90's, such techniques are not easy to come by, and the fine-grained restrictions on the reductions can only make matters worse.

Two notable success stories, highlighted in a recent survey by Rubinstein and Vassilevska Williams [250], are the Distributed PCP framework [11] based on algebraic error-correcting codes that has lead to strong results for many pair-finding type of problems [1, 111–113, 195, 196, 249], and a graph-products technique [44] that has lead to impressive inapproximability results for computing the diameter of a graph [83, 84, 125, 126, 128, 210]. Nevertheless, we are still far away from satisfactory results for many problems (see the open questions in [11, 250]). Even *distance computations in graphs*, an extensively studied subject in fine-grained complexity, exhibits many huge gaps.

As a case in point, consider the open questions below for three of the most basic problems in the area, each of them with a long list of upper bounds spanning several decades: *distance oracles* [38, 65, 68, 115, 116, 131, 232, 233, 261, 264, 265], *dynamic shortest paths* [67, 74, 75, 108, 130, 153, 175, 176, 246, 258], and *shortest cycle* (girth) [123, 133, 186, 190, 212, 243].

Open Question 1.2.1 (Approximate Distance Oracles)**.** Can we preprocess a graph in $m^{1+o(1)}$ time and answer shortest path queries in $m^{o(1)}$ time with $O(1)$ stretch?

No known constant-approximation algorithm can achieve the desirable time bounds in the open question. The above references take $m^{1+\Theta(1/k)}$ preprocessing time to answer queries with a k-approximation in $m^{o(1)}$ time. Meanwhile, the best conditional lower bound by Pătraşcu, Roditty, and Thorup [233] only rules out a $(3-\varepsilon)$-approximation with such time bounds under a set-intersection conjecture.[2] Existing inapproximability results higher than the $3-\varepsilon$ barrier are either information-theoretic incompressibility arguments [86, 219, 265] and therefore

[1]Similar to saying that the NP-hardness of 3-coloring implies a $4/3-\varepsilon$-hardness of approximation for the chromatic number.

[2]Their conjecture and hardness result apply even for preprocessing algorithms with $m^{1+o(1)}$ space (and unbounded time), but higher lower bounds are not known even when restricting the time complexity.

only rule out $o(m)$ space bounds, or in the cell-probe model [261] and therefore only apply for query times up to $\log n$.[3]

Open Question 2.1.1 (Dynamic Shortest Paths)**.** Can we preprocess a graph in $O(mn)$ time, then support edge-updates in $m^{o(1)}$ amortized time, and answer shortest path queries in $m^{o(1)}$ time with an $O(1)$-approximation?

Again, no known constant factor approximation meets these desirable requirements on the update and query times. It is known how to achieve update and query time $O(m^{1/k})$ with approximation factor $O(k)$ in the partially dynamic (deletions only) case [108] and approximation factor $(\log n)^{O(k)}$ in the fully dynamic case [153]. The only conditional lower bounds are for $(2-\varepsilon)$-approximation algorithms and they follow directly from the lower bounds for the *exact* setting [12, 176, 245], where it is shown that distinguishing distance 2 from 4 is hard.[4]

Open Question 2.1.2 (Girth)**.** Can we return an $O(1)$-approximation to the girth (i.e. the length of the shortest cycle) in $m^{1+o(1)}$ time?

The best known approximation with $m^{1+o(1)}$ running time is super-constant; very recently, Kadaria *et al.* [190] obtained an $O(k)$-approximation in $O(m^{1+1/k})$ time. A lower bound for $(4/3-\varepsilon)$-approximation follows from assuming hardness of triangle finding, as deciding if a graph has a triangle is equivalent to distinguishing between girth 3 vs 4. No better lower bound is known.

Trying to answer the above questions negatively by establishing a lower bound leads to a common barrier, known as the *short cycle barrier*, discussed in Section 1.2.2. Overcoming this barrier is related to Open Question 5 in the distributed PCP paper [11], which asks for gap amplification techniques based on conjectures other than SETH. This is because the exact versions of our distance computation problems are not SETH-hard; their hardness arises from reductions from (detecting or) *listing triangles* in a graph—a problem that is hard under the 3SUM or APSP conjectures, but not under SETH. If the hard instances for triangle finding include

[3]The latter is due to the well-known barrier of proving higher unconditional lower bounds for any problem (see [232, 233]). To prove inapproximability even with the more satisfying $m^{o(1)}$ restriction on the query time, we seem to need the conditional lower bounds approach of fine-grained complexity.

[4]The lower bounds hold even against much higher $O(n^{1-\varepsilon})$ update and query times, but inapproximability results with higher multiplicative factors are not known even if we demand $m^{o(1)}$ update and query times.

short cycle-free graphs, it would imply a negative answer to the open questions above. Hence, understanding these questions boils down to the following open question.

Open Question 2.1.3 (Main Open Question)**.** Can we prove hardness for finding a triangle in a 4-cycle free graph? What if it is k-cycle free for all $4 \leqslant k \leqslant O(1)$?

Any progress on Question 2.1.3 carries over to progress on the aforementioned three open questions, by standard reductions. But it is far from clear why such a gap amplification should be possible. The needle-in-a-haystack flavor (and intuitive hardness) of triangle finding *stems* from the possibility of a triangle hiding amidst plenty of 4- or 5-cycles. In a 4-cycle free graph no two nodes can have more than one common neighbor; doesn't that restrict the search space by too much?[5]

Clearly, we do not expect the triangle finding problem to remain *equally hard* in k-cycle-free graphs as in general graphs, already because k-cycle-free graphs for a large even k are very sparse. Moreover, one can apply a standard reduction, e.g. the one to distance oracles sketched above, and then use an existing upper bound (e.g. [233]) to find a triangle in $m^{1+O(1/k)}$ time. Therefore, the main open question is whether or not the problem becomes *very easy*: Can we find a triangle in a 4-cycle free graph in linear time? This contemplation touches upon a well-known hole in our understanding of graph problems. Indeed, by a simple reduction, even this latter most restricted form of Question 2.1.3 is at least as hard as resolving one of the most infamous open questions in fine-grained complexity:

Open Question 2.1.4. Can we determine if a graph contains a 4-cycle in $m^{1+o(1)}$ time?

In 1994, Yuster and Zwick [276] put forth the conjecture that one cannot detect a 4-cycle in a graph in subquadratic time. The longstanding upper bound is $O(m^{4/3})$ via a high-degree low-degree argument [30]. The running time can also be bounded by $O(n^2)$ because when $m \geqslant 200 \cdot n^{1.5}$ we can simply output "yes": by the Bondy-Simonovitz Theorem [82], a graph with such density must contain a 4-cycle. Frustratingly, to this date, the field of fine-grained complexity has not

[5]Such high-girth assumptions can indeed reduce the complexity of some problems from almost-quadratic to almost-linear. In particular, in the Orthogonal Vectors problem with dimension $d = n^{o(1)}$ (at the core of the Diameter lower bounds, and many others), if no two vectors can have two common coordinates that are non-zero, there is an $O(nd^2)$ algorithm.

managed to show any hardness for this problem. *"What hope do we have to understand more complex problems if we cannot settle the complexity of this simple one, even conditionally?"*[6]

We give answers to all of the above questions, some full and some partial, based on fine-grained complexity assumptions. It turns out that Triangle (detection or listing) requires super-linear time even when the graph has very few short cycles.

2.1.1 First Result: Removing Most k-Cycles

Our first main result is a fine-grained *self-reduction* for Triangle from worst-case $\sqrt{n}$-degree graphs to graphs with few k'-cycles for all $4 \leqslant k' \leqslant k$. For concreteness, consider the All-Edge version where we want to report for each of the $m = O(n^{1.5})$ edges in the graph whether it is in a triangle. This problem is known to require $n^{2-o(1)}$ time, under the 3-SUM Conjecture [201, 231] or under the APSP Conjecture [268], and this holds even for graphs of maximum degree $\sqrt{n}$. Thus it is a very plausible conjecture that $n^{2-o(1)}$ is required. (See Theorem 2.4.1 and the discussion in Section 2.7.) A worst-case input graph to this problem might have up to $n^{k/2+1/2}$ k-cycles. Given such a graph, for a sufficiently small constant $\alpha > 0$ depending on k, the following theorem constructs many subgraphs such that: (1) solving All-Edge-Triangle on all of these subgraphs suffices to solve the original problem, (2) the total number of edges in all these subgraphs is subquadratic $n^{2-\Omega(1)}$, and (3) the total number of k-cycles in all these subgraphs is subquadratic $n^{2-\Omega(1)}$. The latter implies that a linear-time algorithm for All-Edge-Triangle in graphs with few short cycles implies a subquadratic algorithm for the starting problem and refutes the popular conjectures.

Theorem 2.1.5 (Removing Most k-Cycles)**.** *For any choice of constants $k \geqslant 4, \alpha \in (0, \frac{1}{2})$, and $\varepsilon \in (0, \frac{3-\omega}{4})$ the following holds. Given a graph G with n vertices and maximum degree at most $\sqrt{n}$, there is a randomized algorithm, running in time $O(n^{2-\varepsilon})$, that returns a subset of the edges $E' \subseteq E(G)$ and a collection of $s = n^{3/2-3\alpha}$ subgraphs $G_1, \ldots, G_s \subseteq G$ such that:*

- *Every edge $e \in E'$ participates in a triangle of G.*

[6]This is a quote from the survey by Rubinstein and Vassilevska Williams [250] where it is referring to the approximability of the graph diameter problem. We find it no less poignant when considering the 4-cycle problem.

- *If an edge $e \in E(G)$ participates in a triangle of G, then it is either in E' or it participates in a triangle in at least one subgraph G_i.*
- *With high probability, each G_i has $O(n^{1/2+\alpha})$ vertices and maximum degree $O(n^{\alpha})$.*
- *For every i, the expected total number of k'-cycles of sizes $4 \leqslant k' \leqslant k$ in G_i is at most $O(n^{\frac{\omega-1}{4}+k\alpha+\varepsilon})$.*

This result achieves a weaker statement than that asked by Question 2.1.3 because it does not remove all short cycles. Still, it is sufficient for fully resolving Questions 1.2.1 and 2.1.1 above. Intuitively, by applying the standard reductions (as described above), each of the few remaining short cycles might result in a false positive: a pair $\{b, c\}$ that has short distance even though it is not in a triangle. But since the number of such cycles is small (and the degrees in the G_i graphs are small), they can all be filtered in a post-processing stage in subquadratic time.

Before giving the inapproximabilty results, let us briefly explain why the matrix multiplication exponent $2 \leqslant \omega < 2.37286$ [19] appears in our statements. Perhaps counter-intuitively, our lower bounds get *higher* the closer ω gets to 2. Roughly speaking, this is because our results follow from reductions that employ several procedures, including fast matrix multiplication, to extract these subgraphs with few short cycles from a given graph. In any case, our results are new and meaningful for any $2 \leqslant \omega < 2.37286$ (or even any $2 \leqslant \omega < 3$); the only difference is in the constants.

Applications Our first corollary improves the ≈ 3 hardness of Pătraşcu, Roditty, and Thorup [233] all the way up to $\omega(1)$, showing that $O(k)$-approximation with $O(m^{1+1/k})$ preprocessing is indeed the right tradeoff for distance oracles with $O(m^{1/k})$ query time. Our lower bound is comparable to that of Sommer, Verbin, and Yu [261] in the cell-probe model, except that we allow much higher query time: $m^{\Omega(1)}$ vs. their $O(1)$. Moreover, our lower bound applies to the easier *offline* version of the problem where all the queries are given in advance; previous lower bounds [232, 233, 261] do not apply to this restricted setting.[7] If $\omega = 2$ our lower bound becomes $m^{1+\frac{1}{4k-2}-o(1)}$ time for a $(k-\delta)$-approximation; with the current ω it is $\Omega(m^{1+\frac{1}{6.3776k-4.3777}})$.

[7] In the stronger models that these papers consider, where we measure space/probes rather than time, this offline problem becomes trivially easy.

Corollary 2.1.6 (Hardness of Approximation for Offline Distance Oracles)**.** *Let $k \geqslant 4$ be an integer, and let $\varepsilon, \delta > 0$. Define $c = \frac{4}{3-\omega}$ and $d = \frac{2\omega-2}{3-\omega}$. Assuming either the 3-SUM Conjecture or the APSP Conjecture, no algorithm can return a $(k-\delta)$-approximation to the distance between m pairs of nodes in a simple graph with m edges in time $O(m^{1+\frac{1}{ck-d}-\varepsilon})$. Consequently, there is no $(k-\delta)$-approximate distance oracle with $O(m^{1+\frac{1}{ck-d}-\varepsilon})$ preprocessing and $O(m^{\frac{1}{ck-d}-\varepsilon})$ query time.*

The *Offline Distance Oracle* problem in the above corollary is at the core of the dynamic shortest paths problem as well. By a straightforward reduction it implies that Chechik's decremental APSP k-approximation algorithm in total time $O(m^{1+1/\alpha k})$ is tight up to the constant α. For fully dynamic algorithms, we can strengthen the result further by ruling out algorithms that start with a cubic-time preprocessing phase (which is natural as it gives the algorithm enough time to precompute all the distances). However, in this fully dynamic case, the best known upper bound only achieves a $(\log n)^{O(k)}$-approximation in $O(m^{1+1/k})$ time.

Corollary 2.1.7 (Hardness of Approximation for Dynamic APSP)**.** *Let $k \geqslant 4$ be an integer, and let $\varepsilon, \delta > 0$, $c = \frac{4}{3-\omega}$ and $d = \frac{2\omega-2}{3-\omega}$. Assuming either the 3-SUM Conjecture or the APSP Conjecture:*

- *No algorithm can maintain a simple graph through a sequence of edge-deletion updates in a total of $O(m^{1+\frac{1}{ck-d}-\varepsilon})$ time, while answering distance queries between a given pair of nodes with a $(k-\delta)$-approximation in $O(m^{\frac{1}{ck-d}-\varepsilon})$ time.*
- *No algorithm can preprocess a simple graph in $O(n^3)$ time and then support (fully dynamic) updates and queries in $O(m^{\frac{1}{ck-d}-\varepsilon})$ time, where an answer to a query is a $(k-\delta)$-approximation to the distance between a given pair of nodes.*

We next go back to the 4-Cycle problem. A direct corollary of our theorem is that the All-Edge version has a super-linear lower bound, finally extending the $m^{3/2}$ lower bound for triangle enumeration from Pătraşcu's seminal paper [231] to a hardness result for 4-cycle enumeration.[8] If $\omega = 2$ the lower bound is $m^{5/4-o(1)}$, and with the current ω it is $\Omega(m^{1.1927})$.

[8]Pătraşcu's lower bound is presented as a lower bound for the *listing* problem, rather than enumeration, where we are required to list m triangles (and the lower bound is $m^{4/3-o(1)}$). Our lower bound also extends to this version, but the exponent is smaller.

Corollary 2.1.8 (Hardness for k-Cycle Enumeration). *Let $k \geqslant 3$ be an integer, and let $\varepsilon > 0$. Assuming either the 3-SUM Conjecture or the APSP Conjecture, no algorithm can process an m-edge graph in $O(m^{1+\frac{3-\omega}{2(4-\omega)}-\varepsilon})$ time and then enumerate k-cycles with $m^{o(1)}$ delay.*

Unlike our two previous corollaries, the lower bound in Corollary 2.1.8 does not get weaker with k. This is because for all $k > 4$ we can simply apply Theorem 2.1.5 with $k = 4$ and then use known simple gadget reductions that show that k-cycle (detection or enumeration) for any k is at least as hard as either the $k = 3$ or $k = 4$ case (see [122]).[9] Either way, we separate k-cycle from the class DELAYC$_{lin}$ of problems solvable with linear time preprocessing and constant delay [137]. This class has received significant attention in recent years from the enumeration algorithms community (see e.g. [96,97,152,255]). Our result is somewhat surprising because enumerating *all* cycles (without restricting k) is *in* the class DELAYC$_{lin}$ [78].

Theorem 2.1.5 does not fully resolve the Main Open Question 2.1.3, because it does leave $n^{\Omega(1)}$ short cycles in the graph. This is not an issue for all of the applications above because the application problem returns multiple answers (that can be filtered afterwards). Unfortunately, this cannot be done for problems with a single output, such as our most basic 4-Cycle detection problem. Nonetheless, with a bit more work we can actually get rid of all k-cycles in the $k = 4$ case, as we discuss next.

2.1.2 Second Result: Removing *all* 4-Cycles

Our most technical result is a strengthening of Theorem 2.1.5, giving a reduction to graphs that are *completely* 4-cycle-free. The following theorem is analogous to Theorem 2.1.5 and should be thought of in the same way; as a self-reduction from Triangle. The main two differences are that it only works for $k = 4$, but it achieves the much stronger property of 4-cycle freeness in the subgraphs it produces.

Theorem 2.1.9. *For any choice of constant $\alpha \in (0, \frac{3-\omega}{8})$ and $\varepsilon \in (0, \frac{3-\omega}{4} - 2\alpha)$ the following holds. Given a graph G with n vertices and maximum degree at most $\sqrt{n}$, there*

[9]The reduction simply subdivides some of the edges. Note that this trick is not useful in the hardness of approximation context because subdividing edges increases the distances even in the yes-case.

is a randomized algorithm, running in time $O(n^{2-\varepsilon})$, that returns a subset of the edges $E' \subseteq E(G)$ and a collection of $s = n^{3/2-3\alpha}$ subgraphs $G_1, \dots, G_s \subseteq G$ such that:

- *Every edge $e \in E'$ participates in a triangle of G.*
- *If an edge $e \in E(G)$ participates in a triangle of G, then it is either in E' or it participates in a triangle in at least one subgraph G_i.*
- *With high probability, each G_i has $O(n^{1/2+\alpha})$ vertices and maximum degree $O(n^{\alpha})$.*
- *With probability larger than 0.99, no subgraph G_i contains a 4-cycle.*

This result fully resolves the main Open Question 2.1.3 in the $k = 4$ case. Consequently, we improve the lower bound for girth approximation in $m^{1+o(1)}$ time from $4/3 - \delta$ to $5/3 - \delta$; thus making the first non-trivial step towards Open Question 2.1.2. And most importantly, we establish the first conditional lower bound for 4-Cycle detection, resolving Open Question 2.1.4. If $\omega = 2$ the lower bound is $m^{7/6-o(1)}$, and with the current ω it is $\Omega(m^{1.1194})$. Note, however, that Corollary 2.1.10 uses a less standard conjecture compared to our previous results.

Corollary 2.1.10 (Hardness for Triangle in 4-Cycle-Free Graphs)**.** *Assuming that triangle detection in graphs with maximum degree at most $\sqrt{n}$ requires $n^{2-o(1)}$ time, no algorithm can solve any of the following problems in $O(m^{1+\frac{3-\omega}{2(5-\omega)}-\varepsilon})$ time, for any $\varepsilon > 0$:*

- *Decide if an m-edge graph has a 4-cycle.*
- *Decide if an m-edge 4-cycle-free graph has a triangle.*
- *Compute a $(5/3 - \delta)$-approximation to the girth of an m-edge graph, for any $\delta > 0$.*

As mentioned already, folklore gadget reductions show that either Triangle or 4-Cycle can be reduced to a single instance of k-Cycle detection on the same number of edges, for any $k \geqslant 3$ (we add a proof of this statement in Appendix 2.8.1, similar reductions appear, for example, in [122]). Thus, we establish a super-linear $\Omega(m^{1.1194})$ lower bound for k-Cycle detection for *all* constant $k \geqslant 3$.

Breaking the hardness assumption at the base of our conditional lower bound would be a major breakthrough. The longstanding upper bound for triangle detection is $O(\min\{m^{2\omega/(\omega+1)}, n^{\omega}\})$ [30]. Even if an optimal matrix multiplication algorithm exists ($\omega = 2$) no algorithm breaks the quadratic barrier when $m = n^{1.5}$.

This continues to be the case in the natural setting where the maximum degree (rather than the average degree) is $\sqrt{n}$. Note that we cannot base these lower bounds on 3SUM or APSP, as we did in the results above for problems with multiple outputs, until we know how to base the hardness of Triangle detection itself on these assumptions. See Section 2.7 for further discussion.

2.2 Technical Overview

The goal of this section is to give an overview of the main new ideas that go into our short cycle removal technique. As discussed in the introduction, the technical barriers are most prominently apparent in the challenge of proving a hardness result for 4-Cycle (detection). For this reason, we choose to focus this section on this result, giving a tour of the reduction from Triangle (detection) to 4-Cycle. All of our conceptually new ideas go into this result and can be appreciated more clearly in this simple context. Afterwards, in Section 2.2.2 we point out how these ideas lead to Theorems 2.1.5 and 2.1.9. The additional required ideas are either standard tricks (e.g. for the applications) or technical but unsurprising generalizations (e.g. for the $k > 4$ case); we briefly mention them in Section 2.2.2. While this presentation of results goes in the reverse order to that of the introduction, it has the advantage of presenting all important ideas while the reader needs to only think about the following deceptively simple goal: *solve Triangle in $\sqrt{n}$-degree graphs in subquadratic time, given a linear-time algorithm for 4-Cycle.*

Some notation: We assume that the input graph $G = (V, E)$ for the triangle finding and all-edge triangle problems is tripartite with sides A, B, and C.[10] We denote the set of neighbors of a node u by $N(u)$. We use $[k]$ to denote the set of integers $\{1, 2, \cdots, k\}$, and we use the notation "$\{4, .., k\}$-cycles" to denote the set of cycles of length at least 4 and at most k. We say that an event happens with high probability if it happens with probability at least $1 - 1/n^c$, for an arbitrarily large constant c. We denote by ω the matrix multiplication exponent. The term "subquadratic" refers any bound of the form $O(n^{2-\varepsilon})$, for any constant $\varepsilon > 0$. Moreover, we always treat k as a constant.

[10] If not, let $A = B = C = V$ and copy each edge in E three times.

2.2.1 Triangle to 4-Cycle

To reduce Triangle to 4-Cycle, we want to convert a hard instance for Triangle in a way such that any triangle becomes a 4-cycle. Since a hard instance for Triangle is a tripartite graph with sides A, B, and C, perhaps the first idea that one may try to use is to subdivide the edges between B and C, by adding a dummy node bc on each $\{b, c\}$ edge. Indeed, if the original graph doesn't contain a 4-cycle, then a 4-cycle in the new graph must use a dummy node, and therefore, the existence of a 4-cycle in the new graph implies the existence of a triangle in the original graph.

However, the original graph may have up to $n^{2.5}$ 4-cycles.[11] In particular, even after adding the dummy nodes between B and C, there could still be up to $n^{2.5}$ 4-cycles between A and B (4-cycles that use two nodes from A and two nodes from B), up to $n^{2.5}$ 4-cycles between A and C, and up to $n^{2.5}$ 4-cycles that use two nodes from A and one node from each of B and C. None of these 4-cycles uses a dummy node, and therefore, the existence of a 4-cycle in the new graph doesn't necessarily imply the existence of a triangle in the original graph. We call these *false* 4-*cycles*. Notably, this issue does not arise when reducing to k-cycle detection for odd k (see [122]) and it can be side-stepped easily in harder contexts such as the directed[12] or counting[13] versions or in other models [8,45][14]. Alas, such simple tricks do not work in the most basic case. As discussed in the introduction, this is not a mere technicality but *the* obstacle for gap amplification results.

To overcome this, one may try to remove all the 4-cycles from the graph before applying the reduction, perhaps by finding a set of edges that intersects all of them, checking whether there is a triangle that uses one of these edges, and then removing these edges from the graph. Indeed, this would leave the graph without any 4-cycles. However, even if we can efficiently check for each of these edges whether it is in a triangle, finding $n^{2.5}$ 4-cycles is impossible in subquadratic time.

Hitting Cycles Faster Than Triangles: Random Slicing At a high-level, our first main idea is simple: since a 4-cycle uses one more node than a triangle, a random subsampling reduces the number of 4-cycles *at a higher rate* than it reduces the

[11] This is because each edge may be in up to n 4-cycles.

[12] Where the directions can prevent the existence of false cycles.

[13] By counting the number of 4-cycles in the induced graph on subsets of the three parts we can find out the exact number of false 4-cycles and then subtract it from the total number.

[14] E.g. in databases a cycle can be forced *by definition* to use one node from each of the three parts and one dummy node.

number of triangles. Indeed, suppose that we subsample nodes, leaving each node in the graph with probability p. A triangle survives with probability p^3 while a 4-cycle only survives with a smaller probability p^4. To implement this idea we use the following *random slicing* approach.

Roughly speaking, instead of reducing Triangle to 4-Cycle in the original tripartite graph, we break the graph into triangle-disjoint tripartite subgraphs,[15] which we refer to as *slices*, and reduce Triangle to 4-Cycle in each of these slices. This way, we would only need to remove 4-cycles that are fully contained in the slices, and not 4-cycles *across* the slices. In more detail, we partition each of A, B and C randomly into $n^{1/2-\alpha}$ sets, each of size $n^{1/2+\alpha}$ (where each node joins one of the sets uniformly at random). In order to solve Triangle in the original graph, it suffices to solve Triangle in each (A_j, B_k, C_ℓ) slice, for $j, k, \ell \in [n^{1/2-\alpha}]$. For each such slice, we want to reduce Triangle to 4-Cycle by first removing all the 4-cycles in the slice, and then subdividing the edges between B_k and C_ℓ. This time, the expected total number of 4-cycles in all the slices is smaller than $n^{2.5}$, for $\alpha < 1/2$. This is because each slice is expected to have $n^{1/2+4\alpha}$ 4-cycles, and in total, over all slices, we have $n^{3/2-3\alpha+1/2+4\alpha} = n^{2+\alpha}$ 4-cycles in expectation.

Unfortunately, $n^{2+\alpha}$ 4-cycles is still too much. If the number of 4-cycles is super-quadratic, then finding and removing all of them in subquadratic time is hopeless. In other words, while we *can* hit 4-cycles at a higher rate than we hit triangles with random slicing, this higher rate is not fast enough to get the number of 4-cycles down from the worst-case $n^{2.5}$ to subquadratic without essentially deleting all edges.

Nevertheless, observe that if we started with fewer than $n^{2.5}$ 4-cycles in the original graph, then the expected number of 4-cycles over all the slices following the random slicing *would be* subquadratic. Can we prove that graphs with too many 4-cycles are not actually hard for Triangle?[16]

Structure vs. Randomness: Dense-Piece Removal Our next main observation is that random $\sqrt{n}$-regular graphs only have $O(n^2)$ 4-cycles. So if a graph has the worst-case $n^{2.5}$ number of 4-cycles, then it must have a lot of structure that

[15]By triangle disjoint we mean that each triangle appears in exactly one of these tripartite subgraphs.

[16]It is natural but perhaps a bit surprising that our goal switched from proving the hardness of Triangle in 4-cycle free graphs to showing the easiness of Triangle in graphs with a maximal number of 4-cycles.

one could potentially exploit for solving Triangle faster. A priori, this may not sound like a promising approach because we know that Triangle is very easy in random graphs,[17] and its hardness arises from structure that existing algorithms cannot exploit. Fortunately, we identify a connection between the existence of many 4-cycles and the existence of dense subgraphs that we do know how to exploit algorithmically. This is based on the fact that the savings from using fast matrix multiplication for Triangle are greater in denser graphs. This is a novel use of the structure vs. randomness paradigm [263] in the context of Triangle.[18]

In a bit more detail, we show that in subquadratic time we can find a set of edges $\tilde{E}$, with an answer to each edge in the graph to whether it participates in a triangle containing an edge in $\tilde{E}$, such that the graph induced by $E \setminus \tilde{E}$ has fewer than $n^{2+\gamma}$ 4-cycles, for some $0 \leqslant \gamma < 1/2$ to be chosen later. This is based on the connection between 4-cycles and dense subgraphs. One can show that an n-node graph with maximum degree $\sqrt{n}$ and at least $n^{2+\gamma}$ 4-cycles must contain at least $n^{1+\gamma}$ dense subgraphs, each with $2\sqrt{n}$ nodes and roughly $n^{1/2+\gamma}$ edges.[19] We refer to such subgraphs as *dense pieces*. In particular, each of these dense pieces lies between two neighborhoods $N(u)$ and $N(v)$ where $\{u, v\}$ is an edge. We can use this property to find a dense piece efficiently: we sample $\approx n^{1/2-\gamma}$ edges $\{u, v\}$, and for each of them we sample $\approx n^{1/2-\gamma}$ pairs of nodes between $N(u)$ and $N(v)$ to estimate the number of edges in $N(u) \times N(v)$. With high probability, for one of the sampled edges $\{u, v\}$, there are at least $\approx n^{1/2+\gamma}$ edges between $N(u)$ and $N(v)$, and it will be detected when we estimate the number of edges between $N(u)$ and $N(v)$. After finding the dense piece $N(u) \cup N(v)$, we use a matrix multiplication approach, together with a high-degree low-degree analysis, to efficiently check whether there is a triangle that uses an edge from the dense piece. Hence, by removing all the dense pieces gradually, where in each step we find a new dense piece, check whether there is a triangle that uses an edge from the dense piece, and then remove it, we obtain a graph with fewer than $n^{2+\gamma}$ 4-cycles. Since in each step we remove a dense piece of $\Omega(n^{1/2+\gamma})$ edges, and the number of edges is $O(n^{3/2})$, the number of steps is bounded by $O(n^{1-\gamma})$. For

[17]In a random $\sqrt{n}$-regular graph, any edge is in a triangle with constant probability, so we can find a triangle in $O(\sqrt{n})$ expected time.

[18]The other two examples that come to mind are the mildly subcubic combinatorial algorithm of Bansal and Williams [54] using the Freize-Kannan regularity lemma, and the distributed algorithms (see [107]) that exploit an expander decomposition of the graph.

[19]This is because there are $n^{3/2}$ edges, and each edge participates in at most n 4-cycles.

an appropriate choice of $\gamma = \frac{\omega-1}{4} + \varepsilon' < 0.345$, for an arbitrarily small constant $\varepsilon' > 0$, we show that the total running time for removing all the dense pieces is subquadratic.

Hence, we first remove all dense pieces in subquadratic time, which leaves only $n^{2+\gamma}$ 4-cycles to begin with. Then, we apply the random slicing, and reduce Triangle to 4-Cycle in each of the (A_j, B_k, C_ℓ) slices where we know that the total number of false 4-cycles is subquadratic. This brings us to the final task of removing all of the remaining 4-cycles in subquadratic time. Let us remark at this point that for Theorem 2.1.5 and its applications where we can tolerate the existence of few cycles, this extra step is not necessary.

Output-Sensitive False Cycle Removal Following the dense-piece removal and the random slicing, the total number of 4-cycles in all the slices is $n^{3/2+\gamma+\alpha}$, which for some choice of $\alpha < 0.155$, is subquadratic. It may still seem challenging to list all of them efficiently, even when their number is subquadratic: we do not even know how to find *one* 4-cycle in a general $\sqrt{n}$-degree graph in subquadratic time.

By exploiting a special property of 4-cycles in tripartite graphs, together with the small degree property of each slice, we design an algorithm that lists all false cycles in time that is linear in their number. The crux of our idea is that all the 4-cycles in a slice can be found by looking *only at one part of the slice at a time*. For instance, to list all the 4-cycles that use two nodes from A_j, it suffices to list all the $\{a, u, a'\}$ two-paths in the slice, where $\{a, a'\} \subseteq A_j$. For this, we can list all the two-paths between A_j and B_k, and all the two-paths between A_j and C_ℓ, and then find all the 4-cycles that use a pair $\{a, a'\} \subseteq A_j$ (by looking at all the two-paths that the pair $\{a, a'\}$ participates in). At first sight, since we have $n^{3/2-3\alpha}$ slices, each containing $n^{1/2+3\alpha}$ pairs $\{a, a'\}$ that can be connected by a two-path, this approach may seem to take quadratic time inevitably. Yet, with an additional trick, and a more sophisticated *global* analysis that takes into account all the slices at once, we are able to charge the running time to the total number of 4-cycles in all the slices, which is subquadratic.

As a final clean-up step before subdviding the edges between B_k and C_ℓ and making a call to the 4-Cycle algorithm, we delete an edge from each of the $n^{3/2+\gamma+\alpha}$ 4-cycles that were found. But first, we must check whether any of them is in a triangle, and we cannot afford to spend the trivial $\sqrt{n}$ time for each. Fortunately, we only need to look for a triangle *in the slice* where nodes only have degree n^α, making the total time $n^{3/2+\gamma+2\alpha}$ still subquadratic for $\alpha < 0.0775$.

Thus, in subquadratic time we can make sure that all calls that we make to the 4-Cycle algorithm are made on graphs without false 4-cycles. To conclude, we point out that the total sizes of all the 4-Cycle instances that our reduction produces is also subquadratic, so if we could solve 4-Cycle in linear time (or even $m^{1.01}$) we would get a subquadratic algorithm for Triangle. Indeed, the number of slices is $n^{3/2-3\alpha}$ and each has $n^{1/2+2\alpha}$ edges.

2.2.2 The Theorems and Corollaries

First, let us clarify the connection between the above reduction and our Theorems for the $k = 4$ case. The slices are precisely the G_i subgraphs with few cycles that our theorems produce, and the set of edges E' are those edges that we identify (and remove) in the dense-piece removal process as participating in a triangle. For Theorem 2.1.5 the final process of removing all false 4-cycles is not necessary; the number of remaining 4-cycles in the slices is small enough. For Theorem 2.1.9 we do list all remaining 4-cycles and remove an edge from each, while placing it in E' if it participates in a triangle. The slices that result after this clean-up are the 4-cycle free G_i subgraphs that we return.

The $k > 4$ Case To remove most k'-cycles for all $4 \leqslant k' \leqslant k$ we follow a similar route. Even though the number of k'-cycles in a worst-case (or random) graph becomes larger as k' grows, the random slicing method also becomes more effective at reducing their number. The intuition is that a $k' > 4$ cycle uses even more nodes than a triangle does, and so it is even less likely to survive a random subsampling; i.e. longer cycles can be hit at an even higher rate. This leads to the same situation where we would be done if the number of k'-cycles was lower than the worst case, e.g. if the graph was random. With more careful combinatorial arguments we manage to obtain a similar structure vs. randomness result: if a graph has too many k'-cycles then it must have a dense subgraph, and moreover such dense subgraphs can be found efficiently. Once we have that, reducing the number of k'-cycles by removing the dense pieces proceeds in exactly the same way as in the $k = 4$ case.

Roughly speaking, we show that if a graph with maximum degree $\sqrt{n}$ has at least $n^{k'/2+\gamma}$ k'-cycles, then it must contain many $O(\sqrt{n})$-node dense subgraphs (pieces), each with at least $\Omega(n^{1/2+\gamma})$ edges. In particular, each of these dense pieces lies between two neighborhoods of a pair of nodes $\{u, w\}$ that are connected by a simple $(k'-2)$-path (a path of $k'-2$ nodes, including u and w). In more detail, for a simple path of $k'-2$ nodes, $\{u, u_1, \cdots, u_{k'-4}, w\}$, we say that it is γ-dense if the number of k'-cycles that use it is $\Omega(n^{1/2+\gamma})$. Thus, a γ-dense path implies that the number of edges between $N(u)$ and $N(w)$ is $\Omega(n^{1/2+\gamma})$. We show that if a graph with maximum degree $\sqrt{n}$ has at least $n^{k'/2+\gamma}$ k'-cycles, then it must contain at least $\frac{1}{2}n^{k'/2-1+\gamma}$ simple $(k'-2)$-paths that are γ-dense. Since there are at most $n^{k'/2-1/2}$ $(k'-2)$-paths in the graph, we can use this property to find a dense piece efficiently, by sampling paths and estimating the density between the neighborhoods of the extreme nodes. Similar path counting arguments were also employed in the cell-probe lower bound of Sommer, Verbin, and Yu [261], but the overall argument and set-up is completely different.

This suffices for proving Theorem 2.1.5. Substantially new ideas are required if one wishes to remove *all* k'-cycles, extending Theorem 2.1.9 to $k > 4$, because our output-sensitive enumeration strategy no longer applies.

The Applications The corollaries follow from Theorems 2.1.5 and 2.1.9 in rather standard ways, as suggested in the introduction. See Sections 2.5.3 and 2.6.2 for the full details.

A road-map for the technical parts: In Section 2.5 we prove Theorem 2.1.5, as well as its hardness consequences for distance oracles, dynamic APSP, and k-cycle enumeration. In Section 2.6, we prove Theorem 2.1.9, as well as its hardness consequences for triangle finding in 4-cycle free graphs, 4-cycle finding, and girth approximation.

2.3 Further Related Work

Many previous works derive consequences from high-girth graphs for distance computation problems. For example, in lower bounds for graph sparsification (spanners) [31,238] or compression (distance oracles) [86,219,261,265], and for the number of rounds in a distributed setting [132,170] the lower bound constructions are built on constructions of a high-girth graph, either explicit (see [182]) or hypothetical under the Girth Conjecture [142]. Unfortunately, no one has managed to make such an approach work for conditional lower bounds in P, because such constructions are *too structured* to be worst-case graphs and cannot encode a hard worst-case instance of another problem such as 3SUM. Our approach is diametrically opposed: we start from a worst-case graph and *efficiently* turn it into an (almost) high-girth graph by our short cycle removal technique (albeit with worse parameters than the best explicit constructions).

Countless papers in fine-grained complexity and graph algorithms study triangles and short cycles. Let us mention a few that are more relevant for this work. Roditty and Vassilevska Williams [243] proved a conditional lower bound for a particular approach for approximating the girth of a graph, but left proving hardness (for any algorithm) as a main open question. Dahlgaard, Knudsen, and Stöckel [122] prove the hardness of k-cycle detection for all $k \geqslant 5$ assuming the hardness of the $k = 3$ and $k = 4$ cases. Dudek and Gawrychowski prove that counting 4-cycles is equivalent to computing the quartet distance on trees [134] and to counting 4-patterns in permutations [135]. Unlike in undirected graphs where the k-Cycle detection problem tends to become easier as k grows (because the graph gets sparser), this is not the case in directed graphs where it is conjectured that $n^{2-o(1)}$ is required for large enough $k \geqslant 3$, and this conjecture is implied by other conjectures on the k-Clique problem [14, 211].

Besides the two general techniques mentioned above for hardness of approximation in P, and their applications, there are also results that follow from problem-specific tweaks to the exact-lower-bound constructions. In the context of distance computations in graphs, some examples are for APSP [18, 131], Diameter and related problems (without the use of graph-products) [13, 109, 244], for the Girth in directed graphs [127] (see [110, 227] for recent upper bounds for this problem), and for dynamic near-additive spanners [73]. A more general result talks about the possibility of avoiding the $\log W$ factor that comes from the standard *scaling* trick in $(1 + \varepsilon)$-approximations for a problem with weights up to W by reduction to an unweighted problem [91]. Moreover, there is a connection between *deterministic* approximation algorithms and circuit complexity that has lead to strong inapproximability results [2, 10, 112] but the hardness assumptions underlying such barriers are known to be breakable with randomized algorithms.

Finally, we mention that we are not aware of previous papers studying the complexity of problems when the input is restricted to have a high girth, but such questions were already studied in the context of graph spanners [4, 66].

2.4 Preliminaries

Most of our lower bounds rely on the following theorem, which establishes hardness of a triangle finding problem based on either of the standard conjectures about 3-SUM or All Pairs Shortest Paths (for background on these conjectures see e.g. [267, 270]).

Theorem 2.4.1 (All-Edges-Triangle is 3-SUM and APSP hard [268])**.** *Let $\varepsilon > 0$. Assuming the 3-SUM conjecture or the APSP conjecture, no $O(n^{2-\varepsilon})$-time algorithm can answer for each edge whether it participates in a triangle in a given n-node graph with maximum degree at most $\sqrt{n}$.*

For some background on this theorem, we mention that reductions from 3-SUM to triangle listing were initiated by Pătraşcu [231] and further refined in [201]. Recently, Vassilevska Williams and Xu [268] showed that the 3-SUM conjecture can be replaced by the APSP conjecture, obtaining the same result under either of these conjectures. They also showed a variant of this lower bound which replaces triangle listing by asking for every edge whether it is part of a triangle [268, Corollary 3.9]; this variant is more useful in our context and stated above (slightly rephrased).

2.5 Removing Most k-Cycles

In this section we prove Theorem 2.1.5.

Theorem 2.1.5 (Removing Most k-Cycles)**.** *For any choice of constants $k \geqslant 4, \alpha \in (0, \frac{1}{2})$, and $\varepsilon \in (0, \frac{3-\omega}{4})$ the following holds. Given a graph G with n vertices and maximum degree at most $\sqrt{n}$, there is a randomized algorithm, running in time $O(n^{2-\varepsilon})$, that returns a subset of the edges $E' \subseteq E(G)$ and a collection of $s = n^{3/2-3\alpha}$ subgraphs $G_1, \ldots, G_s \subseteq G$ such that:*

- *Every edge $e \in E'$ participates in a triangle of G.*
- *If an edge $e \in E(G)$ participates in a triangle of G, then it is either in E' or it participates in a triangle in at least one subgraph G_i.*
- *With high probability, each G_i has $O(n^{1/2+\alpha})$ vertices and maximum degree $O(n^{\alpha})$.*

- *For every i, the expected total number of k'-cycles of sizes $4 \leqslant k' \leqslant k$ in G_i is at most $O(n^{\frac{\omega-1}{4}+k\alpha+\varepsilon})$.*

The proof of Theorem 2.1.5 is provided in Section 2.5.2. We start with the dense-piece removal step, which is described in Section 2.5.1. Then, in Section 2.5.2, we use a randomized slicing together with the dense-piece removal step to deduce the theorem. In Section 2.5.3, we prove hardness results that are consequences of the theorem.

2.5.1 Dense Piece Removal

In this section we prove the following lemma.

Lemma 2.5.1. *Let $\gamma = (\omega - 1)/4 + \varepsilon$ for an $\varepsilon \in (0, \frac{3-\omega}{4})$, and let $k \geqslant 4$. Given a graph $G = (V, E)$ with maximum degree at most $\sqrt{n}$, there is an $O(n^{2-\varepsilon})$-time algorithm that returns a subset of the edges $\tilde{E} \subseteq E$ and reports all the edges in E that are in a triangle using an edge from $\tilde{E}$, such that the graph $\tilde{G} = (V, E \setminus \tilde{E})$ has at most $O(n^{k/2+\gamma})$ k-cycles.*

That is, Lemma 2.5.1 implies that in order to solve All-Edge Triangle in G in subquadratic time, it suffices to solve All-Edge Triangle in the graph $\tilde{G} = (V, E \setminus \tilde{E})$ in subquadratic time. This is because after reporting all the edges in the graph that are in a triangle using an edge from $\tilde{E}$, it is safe to remove $\tilde{E}$ from the graph (without unintentionally removing triangles with unreported edges) and solve All-Edge Triangle in the obtained graph $\tilde{G}$. The advantage of reducing the problem to solving All-Edge Triangle in $\tilde{G}$ is that $\tilde{G}$ is guaranteed to have significantly less k-cycles compared to a worst-case instance.

A road-map for the proof of Lemma 2.5.1: We start with the useful definition of a γ-dense piece and a γ-dense path (Definition 2.5.2). In Lemma 2.5.3, we show that a graph that has many k-cycles must contain many dense paths, a property that we use in Lemma 2.5.5 to show that a dense subgraph can be found efficiently. In Lemma 2.5.6, we show that given a $\sqrt{n}$-node subgraph, we can check for all edges e in the graph, whether there is a triangle that uses e and an edge from this subgraph efficiently. After the proof of Lemma 2.5.6, we put everything together and present the formal proof of Lemma 2.5.1. Finally, in Theorem 2.5.7, we use the same ideas to show that All-Edge Triangle is 3SUM and APSP hard even when the graph contains a subquadratic number of triangles, a property that we need in one of our applications in Section 2.5.3 (namely, the lower bound for k-cycle enumeration).

Definition 2.5.2 (**γ-Dense Pieces and γ-Dense Paths**)**.** Given an n-node graph with maximum degree at most $\sqrt{n}$, we say that a set of nodes of size at most $2\sqrt{n}$ is a γ-dense piece if the subgraph induced by these nodes has at least $n^{1/2+\gamma}/2$ edges. Furthermore, we say that a simple $(k-2)$-path, $\{u, u_1, \cdots, u_{k-4}, w\}$, is γ-dense if the number of k-cycles that use it is at least $n^{1/2+\gamma}/2$.

Hence, a γ-dense $(k-2)$-path $\{u, \cdots, w\}$ implies that there are $n^{1/2+\gamma}/2$ edges between $N(u)$ and $N(w)$, which implies that $N(u) \cup N(w)$ is a γ-dense piece. In the following lemma, we show that if there are many k-cycles in the graph then there are many γ-dense $(k-2)$-paths, which implies that there are many γ-dense pieces.

Lemma 2.5.3. *Let $k \geqslant 4$. For every $0 \leqslant \gamma < 1/2$, every n-node graph with maximum degree at most $\sqrt{n}$ that has at least $n^{k/2+\gamma}$ k-cycles must contain at least $\frac{1}{2}n^{k/2-1+\gamma}$ simple γ-dense $(k-2)$-paths $\{u, u_1, \cdots, u_{k-4}, w\}$.*

Proof. Let c be the number of k-cycles in the graph, and for a simple path p of $k-2$ nodes, let c_p be the number of k-cycles that use p as a subpath. Observe that

$$c \leqslant \sum_{(k-2)\text{-paths } p} c_p$$

Furthermore, the number of $(k-2)$-paths in the graph is at most $n \cdot (\sqrt{n})^{k-3} = n^{\frac{k-1}{2}}$, because there are n ways to pick the first node in the path, and $(\sqrt{n})^{k-3}$ ways to extend this node to a $(k-2)$-path. Moreover, observe that each $(k-2)$-path, $\{u, u_1, \cdots, u_{k-4}, w\}$, participates in at most n k-cycles, because there are at most n edges between $N(u)$ and $N(w)$. Hence, since each $(k-2)$-path that is not γ-dense participates in at most $\frac{1}{2}n^{1/2+\gamma}$ k-cycles, if we have fewer than $\frac{1}{2}n^{k/2-1+\gamma}$ γ-dense $(k-2)$-paths, this implies that the number of k cycles is bounded by

$$c < \frac{1}{2}n \cdot n^{k/2-1+\gamma} + \frac{1}{2}n^{1/2+\gamma} \cdot n^{\frac{k-1}{2}} = n^{k/2+\gamma},$$

which is a contradiction. □

The remainder of this section is devoted to showing that there is a subquadratic-time algorithm that removes all dense pieces, leaving the obtained graph with fewer than $n^{k/2+\gamma}$ k-cycles by Lemma 2.5.3. We start with the following proposition that follows by a standard Chernoff argument.

Proposition 2.5.4. *Let X and Y be two $\sqrt{n}$-size sets of nodes (not necessarily different). Sample $S = 200n^{1/2-\gamma}\log n$ pairs in $X \times Y$ independently and uniformly at random. It holds that:*

1. *If the number of edges between X and Y is at least $n^{1/2+\gamma}/2$, then at least $50\log n$ of the sampled pairs are edges, with probability at least $1 - 1/n^{10}$.*

2. *If the number of edges between X and Y is smaller than $n^{1/2+\gamma}/200$, then fewer than $50\log n$ of the sampled pairs are edges, with probability at least $1 - 1/n^{10}$.*

In the following lemma, we show that if the graph has at least $n^{k/2+\gamma}$ k-cycles, then a dense subgraph can be found efficiently.

Lemma 2.5.5. *Let $k \geqslant 4$. Given an n-node graph of maximum degree at most $\sqrt{n}$ that contains at least $n^{k/2+\gamma}$ k-cycles, there is an $O(n^{1-2\gamma}\log^2 n)$-time algorithm that finds a pair of nodes $\{u, w\}$, such that the number of edge between $N(u)$ and $N(w)$ is at least $n^{1/2+\gamma}/200$, with high probability.*

Proof. First, we show that we can sample a γ-dense $(k-2)$-path efficiently. Observe that in $O(1)$-time, we can sample a $(k-2)$-path, such that each simple path $\{u, u_1, \cdots, u_{k-4}, w\}$ is sampled with probability at least $1/n^{(k-1)/2}$. This can be done by first sampling a starting node, and in each step we sample a node from the neighborhood of the previously sampled node, until we sample a $(k-2)$-path. Hence, by Lemma 2.5.3, the probability mass of the simple $(k-2)$-paths that are γ-dense is at least $n^{\frac{k}{2}-1+\gamma}/n^{(k-1)/2} = n^{\gamma-1/2}$. Therefore, by sampling $100n^{1/2-\gamma}\log n$ such $(k-2)$-paths, one of them is a simple γ-dense path with high probability. Moreover, for each sampled $(k-2)$-path, $\{u, u_1, \cdots, u_{k-4}, w\}$, we sample $200n^{1/2-\gamma}\log n$ pairs in $N(u) \times N(w)$, and check how many of the sampled pairs are edges. If the number is at least $50\log n$, we output the pair of nodes $\{u, w\}$. By Proposition 2.5.4, the algorithm finds at least one pair $\{u, w\}$ with the desired property with high probability. Furthermore, by Proposition 2.5.4, for any pair $\{u, w\}$ that the algorithm outputs there are at least $n^{1/2+\gamma}/200$ edges between $N(u)$ and $N(w)$, with high probability. $\square$

Next, we show that given two $\sqrt{n}$-size sets of nodes X and Y (not necessarily disjoint), there is an efficient algorithm that reports all the edges in the graph that are in a triangle using an edge from $X \times Y$.

Lemma 2.5.6. *Given an n-node graph with maximum degree at most $\sqrt{n}$, and two $\sqrt{n}$-size sets of nodes X and Y, there is an $O(n^{\frac{3+\omega}{4}})$-time algorithm that finds all the edges in the graph that are in a triangle using an edge from $X \times Y$.*

Proof. Let $\beta > 0$ be a constant to be chosen later, V^h be the set of nodes that have at least $n^{1/2-\beta}$ neighbors in $X \cup Y$, and V^ℓ be the set of nodes that have fewer than $n^{1/2-\beta}$ neighbors in $X \cup Y$. Observe that for any edge e that is in a triangle that uses an edge from $X \times Y$, it holds that either the triangle uses a node from V^ℓ (in which case either $e \in V^\ell \times (X \cup Y)$ or $e \in X \times Y$ and the third node is in V^ℓ), or it uses a node from V^H (in which case either $e \in V^H \times (X \cup Y)$ or $e \in X \times Y$ and the third node is in V^H). We start with the low-degree case, in which we find all the triangles that use a node from V^ℓ.

Low-degree nodes: First, observe that we can find the set of nodes V^ℓ in linear time, as follows. We go over the nodes in $X \cup Y$, and for each of them we mark its neighbors. Then, we go over all the nodes in the graph and take those that were marked fewer than $n^{1/2-\beta}$ times to V^ℓ.

To find the triangles involving nodes in V^ℓ, we go over all the nodes in V^ℓ and for each of them we go over all pairs of neighbors in $X \cup Y$, and for each such pair we check whether it is an edge. To analyse the time complexity of this step, we bucket the set of nodes in V^ℓ by their degrees in $X \cup Y$, where the i'th bucket contains every node $v \in V^\ell$ of degree $|N(v) \cap (X \cup Y)| \in [2^i, 2^{i+1})$, for $0 \leqslant i \leqslant \log(n^{1/2-\beta})$. Observe that the time it takes to process the i'th bucket is $O(\frac{n}{2^i} \cdot (2^{i+1})^2)$. This is because there are $O(n/2^i)$ nodes with degrees in $[2^i, 2^{i+1})$, since the number of edges incident to $X \cup Y$ is $O(n)$. Hence, in total for all buckets, this takes time

$$O\Big(\sum_i \frac{n}{2^i} \cdot (2^{i+1})^2\Big) = O(n \cdot n^{1/2-\beta}) = O(n^{3/2-\beta}).$$

High-degree nodes: Observe that the set V^h can be found in linear time in a similar way that we used to find the set V^ℓ. Furthermore, the size of V^h is at most $O(n^{1/2+\beta})$ since the total number of edges incident to nodes in $X \cup Y$ is at most

$O(n)$. It remains to find all the edges e that are in a triangle that uses a node from V^h and an edge from $X \times Y$. Hence, either $e \in X \times Y$ (in which case we want to check whether it is in a triangle that uses a node from V^h), $e \in V^h \times X$ (in which case we want to check whether it is in a triangle that uses a node from Y), or $e \in V^h \times Y$ (in which case we want to check whether it is in a triangle that uses a node from X). To find these edges, we use a matrix multiplication approach, as follows.

$e \in X \times Y$: To find all the edges in $X \times Y$ that are in a triangle that uses a node from V^h, we use a matrix multiplication algorithm. Consider the Boolean matrices $X \times V^h$ and $V^h \times Y$, where 1-entries indicate edges. By multiplying the two matrices, we get all the pairs $u \in X, w \in Y$ for which there is a 2-path between u and w through V^h. Hence, by going over all the edges in $X \times Y$, we can check for each of them whether it participates in a triangle that uses a node from V^h. Multiplying an $n^{1/2} \times n^{1/2+\beta}$ matrix by an $n^{1/2+\beta} \times n^{1/2}$ matrix takes time $O(n^{w/2} \cdot n^{\beta})$, because we can split it into n^{β} many matrix products on $n^{1/2} \times n^{1/2}$ square matrices.[20] Furthermore, going over all the edges in $X \times Y$ takes $O(n)$ time. Hence, finding all the edges in $X \times Y$ that are in a triangle that uses a node from V^h takes time $O(n^{w/2+\beta})$.

$e \in V^h \times X$ or $e \in V^h \times Y$: To find the edges in $V^h \times X$ that are in a triangle that uses a node from Y, we use a similar matrix multiplication algorithm. This time, consider the matrices $V^h \times Y$ and $Y \times X$, where 1-entries indicate edges. By multiplying the two matrices, we get all the pairs $u \in V^h, x \in X$ for which there is a 2-path between u and x through Y. Hence, by going over all the edges in $V^h \times X$ (which takes $O(n^{1+\beta})$ time), we can check for each of them whether it participates in a triangle that uses a node from Y. Multiplying an $n^{1/2} \times n^{1/2+\beta}$ matrix by an $n^{1/2} \times n^{1/2}$ matrix takes $O(n^{w/2} \cdot n^{\beta})$ time as well.

Finding the edges in in $V^h \times Y$ that are in a triangle that uses a node from X is done in a similar way. Hence, in total, the high degree case takes $O(n^{\omega/2+\beta})$ time.

Putting everything together To optimize the time complexity in total for the high-degree and the low-degree cases, we set $\beta = (3 - \omega)/4$, which implies a

[20]This step could be improved using rectangular matrix multiplication, which for the current value of ω would yield better constants in our lower bounds. Since in the limit for $\omega = 2 + o(1)$ our lower bounds are unaffected, we omit the details.

total running time of $O(n^{(3+\omega)/4})$, as desired. □

Now we are ready to prove Lemma 2.5.1.

Proof of Lemma 2.5.1. We iteratively run the following algorithm that has a 3-step structure: (1) Find a pair $\{u, w\}$ with at least $n^{1/2+\gamma}/200$ edges between $N(u)$ and $N(w)$ by using the algorithm from Lemma 2.5.5 (if the algorithm from Lemma 2.5.5 fails to find such a pair, we know that the graph has fewer than $n^{k/2+\gamma}$ k-cycles, and we stop), (2) report all the edges that are in a triangle that uses an edge from $N(u) \times N(w)$ by using the algorithm from Lemma 2.5.6, and (3) remove all the edges between $N(u)$ and $N(w)$ from the graph. Since the number of edges in the graph is at most $n^{3/2}$, and in each step we remove at least $n^{1/2+\gamma}/200$ edges, the algorithm has at most $O(n^{1-\gamma})$ iterations. Furthermore, in each iteration, step (1) takes $O(n^{1-2\gamma}\log^2 n)$ time, step (2) takes $O(n^{(3+\omega)/4})$ time, and step (3) takes linear time. Hence, in total, the running time is $O(n^{1-\gamma} \cdot n^{(3+\omega)/4})$. For $\gamma = (\omega-1)/4 + \varepsilon$, where $\varepsilon \in (0, \frac{3-\omega}{4})$, this is $O(n^{2-\varepsilon})$, as desired (the upper bound on ε is needed so that we have $\gamma < 1/2$). □

Finally, we finish this section with the following remark and theorem on the number of triangles in the All-Edge Triangle problem.

A remark on the All-Edge Triangle problem: One of our lower bound results in Section 2.5.3 requires the total number of triangles in the All-Edge Triangle instance to be subquadratic, specifically the lower bound for 4-cycle enumeration in Theorem 2.5.8. Interestingly, we can use the same ideas that we presented in this section to show that the All-Edge Triangle problem is still (3-SUM and APSP) hard even when the number of triangles is $O(n^{3/2+\gamma})$, for $\gamma = \frac{\omega-1}{4} + \varepsilon$.

Theorem 2.5.7. *Let $\varepsilon \in (0, \frac{3-\omega}{4})$ and $\gamma = \frac{\omega-1}{4} + \varepsilon$. Assuming the 3-SUM conjecture or the APSP conjecture, no $O(n^{2-\varepsilon})$-time algorithm can answer for each edge whether it participates in a triangle in a given n-node graph with maximum degree at most $\sqrt{n}$, and $O(n^{3/2+\gamma})$ triangles.*

Proof. By Theorem 2.4.1, assuming the 3-SUM conjecture or the APSP conjecture, no $O(n^{2-\varepsilon})$-time algorithm can answer for each edge whether it participates in a triangle in a given n-node graph with maximum degree at most $\sqrt{n}$. Hence, it suffices to show that we can reduce the number of triangles in the input graph to $n^{3/2+\gamma}$ in subquadratic time.

For this, we use the same dense piece removal trick that we introduced in this section. Observe that an n-node graph with at least $n^{3/2+\gamma}$ triangles must contain at least $\frac{1}{2}n^{1/2+\gamma}$ nodes that each participate in at least $\frac{1}{2}n^{1/2+\gamma}$ triangles. Otherwise, since each node can be in at most n triangles, the number of triangles would be smaller than $\frac{1}{2}n \cdot n^{1/2+\gamma} + \frac{1}{2}n^{1/2+\gamma} \cdot n = n^{3/2+\gamma}$, which is a contradiction. Hence, there are at least $\frac{1}{2}n^{1/2+\gamma}$ nodes v for which the number of edges between the nodes in $N(v)$ is at least $\frac{1}{2}n^{1/2+\gamma}$.

Therefore, as long as the number of triangles is at least $n^{3/2+\gamma}$, we can find a $\sqrt{n}$-node subgraph with $\Omega(n^{1/2+\gamma})$ edges efficiently: sample $100n^{1/2-\gamma}\log n$ nodes v, and for each of them sample $200n^{1/2-\gamma}\log n$ pairs in $N(v) \times N(v)$ and check how many of the sampled pairs are edges. If the number of edges is at least $50\log n$, we output $N(v)$. By a similar analysis to the one in Lemma 2.5.5, as long as long as the number of triangles is at least $n^{3/2+\gamma}$, this algorithm finds a $\sqrt{n}$-node subgraph with $\Omega(n^{1/2+\gamma})$ edges, with high probability. Moreover, by Lemma 2.5.6, we report all the edges in the graph that participate in a triangle that uses an edge from the subgraph in $O(n^{\frac{3+\omega}{4}})$ time.

Therefore, by iteratively finding a dense subgraph, checking for each edge in the subgraph whether there is a triangle that uses it, and removing these edges from the graph, we obtain a graph with fewer than $n^{3/2+\gamma}$ triangles. By a similar analysis to the one provided in the proof of Lemma 2.5.1, this takes subquadratic time, as desired. □

2.5.2 Hitting k-Cycles Faster than Triangles: A Proof of Theorem 2.1.5

Proof of Theorem 2.1.5. Let $\gamma = (\omega - 1)/4 + \varepsilon$, where $\varepsilon \in (0, \frac{3-\omega}{4})$. Recall that we can assume without loss of generality that the input graph is tripartite. Let A, B, and C be the three parts. First, we run the algorithm from the dense piece removal step (Lemma 2.5.1) for every $4 \leqslant k' \leqslant k$, and we set E' to be the set of reported edges that are in a triangle that uses an edge from a dense piece $\tilde{E}$. This takes $O(kn^{2-\varepsilon}) = O(n^{2-\varepsilon})$ time. Furthermore, the obtained graph has fewer than $n^{k'/2+\gamma}$ k'-cycles for every $4 \leqslant k' \leqslant k$.

We break the obtained graph into tripartite subgraphs, which we refer to as slices, such that each triangle appears in exactly one slice, as follows. We randomly partition each of the sets A, B and C into $n^{1/2-\alpha}$ sets, each of expected size

$n^{1/2+\alpha}$, where each node joins each of the sets uniformly at random, and independently of the choices for the other nodes. We denote these sets by $\{A_j\}_{j\in[n^{1/2-\alpha}]}$, $\{B_k\}_{k\in[n^{1/2-\alpha}]}$, and $\{C_\ell\}_{\ell\in[n^{1/2-\alpha}]}$. That is[21],

$$A = A_1\dot\cup\dots\dot\cup A_{n^{1/2-\alpha}},\ \ B = B_1\dot\cup\dots\dot\cup B_{n^{1/2-\alpha}},\ \ C = C_1\dot\cup\dots\dot\cup C_{n^{1/2-\alpha}}.$$

By a standard Chernoff argument, for every slice (A_j, B_k, C_ℓ), the number of nodes is $O(n^{1/2+\alpha})$ and the maximum degree is $O(n^{1/2}/n^{1/2-\alpha}) = O(n^\alpha)$ with high probability. It remains to show that the expected number of $\{4,..,k\}$-cycles in each slice is $O(n^{\gamma+\alpha k})$. Observe that the probability that a given k'-cycle is fully contained in the slice (A_j, B_k, C_ℓ) is $1/(n^{1/2-\alpha})^{k'} = 1/(n^{k'/2-k'\alpha})$. Hence, the expected number of k'-cycles that are fully contained in the slice (A_j, B_k, C_ℓ) is $n^{k'/2+\gamma}/(n^{k'/2-k\alpha}) = n^{\gamma+k'\alpha}$. Over all $4 \leqslant k' \leqslant k$, the expected number of $\{4,..,k\}$-cycles is at most $kn^{\gamma+k\alpha} = O(n^{\gamma+k\alpha})$ (since k is constant), as desired. □

2.5.3 Consequences of Theorem 2.1.5

We start with a hardness result for 4-cycle enumeration.

Theorem 2.5.8. *For every $\varepsilon > 0$ there is a $\delta > 0$ such that if there is an algorithm that can preprocess an m-edge graph in $O(m^{1+\frac{3-\omega}{2(4-\omega)}-\varepsilon})$ time and then enumerate 4-cycles with $m^{o(1)}$ delay, then there is an $n^{2-\delta+o(1)}$-time algorithm that given an n-node graph with maximum degree at most $\sqrt{n}$ and $O(n^{2-\delta})$ triangles answers for every edge whether it participates in a triangle with probability at least 9/10.*

Proof. Let G be an n-node tripartite graph with sides A, B, and C. First, we run the subquadratic-time algorithm from Theorem 2.1.5 with ε', α to be chosen later, and $k = 4$. Since the expected number of 4-cycles in each G_i is $O(n^{\frac{\omega-1}{4}+\varepsilon'+4\alpha})$, it follows that the total number of 4-cycles in all the G_i's is at most $O(n^{3/2+\frac{\omega-1}{4}+\alpha+\varepsilon'})$ with probability at least 99/100 (by linearity of expectation and Markov). Furthermore, since each edge that is in a triangle is either in E' or is in a triangle in one of the G_i's, in order to find the remaining edges in $E \setminus E'$ that are in a triangle, it suffices to enumerate the triangles in all the G_i's.

For this, we subdivide the edges in $B \times C$ by adding a dummy node bc on each edge $\{b, c\}$. Hence, any triangle $\{a, b, c\}$ in some G_i becomes a 4-cycle $\{a, b, bc, c\}$;

[21]We use the notation $\dot\cup$ to denote a disjoint union of sets.

these are the only newly introduced 4-cycles. We refer to the other 4-cycles that are not a result of subdividing triangles as *false* 4-cycles; note that each false 4-cycle already was a 4-cycle before subdividing the edges. For each G_i, we run a 4-cycles enumeration algorithm on the subdivided graph. Let $P(m) = m^x$ be the preprocessing time and $D(m) = m^{o(1)}$ be the delay. Since each G_i has $n^{1/2+2\alpha}$ edges with high probability, the total preprocessing time for all G_i's is

$$O(n^{3/2-3\alpha} \cdot P(n^{1/2+2\alpha})) = O(n^{3/2-3\alpha+x(1/2+2\alpha)}),$$

with high probability. For $x < 1 + \alpha/(1/2 + 2\alpha)$ this is subquadratic. On the other hand, the total delay we spend on false 4-cycles is $O(n^{3/2+\frac{\omega-1}{4}+\alpha+\varepsilon'+o(1)})$ with probability at least $99/100$. The remaining delay is spent on enumerating subdivided triangles, and there is only a subquadratic number of them. Hence, for $\alpha = \frac{3-\omega}{4} - \varepsilon''$, for $\varepsilon'' > \varepsilon'$, the total delay is subquadratic. Furthermore, since

$$\begin{aligned} 1 + \alpha/(1/2 + 2\alpha) &= 1 + \frac{\frac{3-\omega}{4} - \varepsilon''}{1/2 + 2(\frac{3-\omega}{4} - \varepsilon'')} \\ &> 1 + \frac{\frac{3-\omega-4\varepsilon''}{4}}{\frac{4-\omega}{2}} \\ &> 1 + (3-\omega)/(2(4-\omega)) - 2\varepsilon'', \end{aligned}$$

when we set $x = 1 + (3-\omega)/(2(4-\omega)) - \varepsilon$, for $\varepsilon = 2\varepsilon''$, the total preprocessing time and the total delay are both subquadratic, as desired. □

Corollary 2.5.9 follows immediately by combining Theorem 2.5.8 and Theorem 2.5.7.

Corollary 2.5.9. *Assuming either the 3-SUM or APSP Conjectures, no algorithm can process an m-edge graph in $O(m^{1+\frac{3-\omega}{2(4-\omega)}-\varepsilon})$ time and then enumerate 4-cycles with $m^{o(1)}$ delay.*

By simple gadget reductions that show that k-cycle (detection, enumeration, or listing) for any k is at least as hard as either the $k = 3$ or $k = 4$ case (see Appendix 2.8) the following corollary follows:

Corollary 2.5.10 (Hardness for k-Cycle Enumeration). *Let $k \geqslant 3$ be an integer, and let $\varepsilon > 0$. Assuming either the 3-SUM Conjecture or the APSP Conjecture, no algorithm* *can process an m-edge graph in $O(m^{1+\frac{3-\omega}{2(4-\omega)}-\varepsilon})$ time and then enumerate k-cycles with $m^{o(1)}$ delay.*

Next, we show a hardness result for Approximate Distance Oracles.

Theorem 2.5.11. *For every $\varepsilon, \delta' > 0$ there is a $\delta > 0$ such that if there is an algorithm running in $O(m^{1+\frac{3-\omega}{2(k+1-\omega)}-\varepsilon})$ time that can $(k/2-\delta')$-approximate the distances between m given pairs of nodes in a given m-edge graph, then there is an $O(n^{2-\delta})$-time algorithm for n-node graphs with maximum degree at most $\sqrt{n}$ that with probability at least $9/10$ answers for every edge whether it participates in a triangle.*

Proof. First, we run the subquadratic-time algorithm from Theorem 2.1.5 with ε' and α to be chosen later. For each of the returned G_i's, we show how to check for every edge in $B \times C$ whether it participates in a triangle. (Checking the edges in $A \times B \cup A \times C$ is symmetric.) For every G_i, we remove the edges in $(B \times C) \cap E(G_i)$, and we denote the obtained graph by G_i'. We run a $(k/2-\delta')$-approximate distance oracle algorithm on G_i', where we query all the $\{b,c\}$ pairs that correspond to the removed edges. We refer to the pairs $\{b,c\}$ for which the algorithm returned an estimate that is smaller than k as the candidates of G_i. For each such candidate pair, we check whether the corresponding edge is in a triangle in G_i, which takes $O(n^{\alpha})$ time per edge. If the edge is found to be in a triangle, we remove it from the set of candidates of G_j for every $j > i$. This ensures that we don't spend too much time on checking whether the same edge is in many different triangles.

Observe that for every edge $\{b,c\}$ that is in a triangle in G_i, the $(k/2-\delta')$-approximation algorithm must return an estimate that is smaller than k when we query the pair $\{b,c\}$, as there is a two-path between b and c in G_i'. Furthermore, for every pair $\{b,c\}$ for which the algorithm returns an estimate that is smaller than k, it holds that there is a path between b and c of length at most $k-1$ in G_i', and therefore the edge $\{b,c\}$ is in a cycle of length at most k in G_i. We refer to the edges $\{b,c\}$ for which the algorithm returns an estimate $< k$ but $\{b,c\}$ is not in a triangle in G_i as *false* edges.

Running time: Let $T(m) = m^x$ be the running time of the distance oracle algorithm (specifically, the total time for preprocessing an m-edge graph and answering m approximate distance queries that are given in advance). In total, running this algorithm for all the G_i''s takes time

$$O(n^{3/2-3\alpha+x(1/2+2\alpha)}).$$

For $x < 1+\alpha/(1/2+2\alpha)$ this is subquadratic. Furthermore, the total number of $\{4,..,k\}$-cycles in all the G_i's is $O(n^{3/2-3\alpha+(\omega-1)/4+\varepsilon'+\alpha k})$ with probability at least $99/100$. Therefore, this is also the total number of times we check whether a false edge is in a triangle, over all the G_i's. For an edge $\{b,c\}$ that participates in a triangle, we run a single check - the first time it was found to be in a triangle in some G_i. Hence, the total running time for this step is $O(n^{3/2+(\omega-1)/4+\varepsilon'+\alpha(k-3)} \cdot n^{\alpha}) = O(n^{3/2+(\omega-1)/4+\varepsilon'+\alpha(k-2)})$. This is subquadratic when we set

$$\alpha = \frac{\frac{1}{2}-\frac{\omega-1}{4}}{k-2} - \varepsilon'' = \frac{3-\omega}{4(k-2)} - \varepsilon''$$

for $\varepsilon'' > \varepsilon'$. For this choice of α, we have that

$$\begin{aligned}
1+\alpha/(1/2+2\alpha) &= 1+\frac{\frac{3-\omega}{4(k-2)}-\varepsilon''}{\frac{1}{2}+2(\frac{3-\omega}{4(k-2)}-\varepsilon'')} \\
&> 1+\frac{\frac{3-\omega-4(k-2)\varepsilon''}{4(k-2)}}{\frac{k+1-\omega}{2(k-2)}} \\
&= 1+\frac{3-\omega}{2(k+1-\omega)}-\frac{4(k-2)\varepsilon''}{2(k+1-\omega)} \\
&> 1+\frac{3-\omega}{2(k+1-\omega)}-\frac{4(k-2)\varepsilon''}{2(k+1-3)} \\
&= 1+\frac{3-\omega}{2(k+1-\omega)}-2\varepsilon''
\end{aligned}$$

Thus, when we set $x = 1+\frac{3-\omega}{2(k+1-\omega)}-\varepsilon$, for $\varepsilon > 2\varepsilon''$, the total running time is subquadratic, as desired. □

Corollary 2.1.6 follows immediately by combining Theorem 2.5.11 with Theorem 2.4.1.

Corollary 2.5.12 (Hardness of Approximation for Offline Distance Oracles). *Let $k \geqslant 4$ be an integer, and let $\varepsilon,\delta > 0$. Define $c = \frac{4}{3-\omega}$ and $d = \frac{2\omega-2}{3-\omega}$. Assuming* *either the* 3*-SUM Conjecture or the APSP Conjecture, no algorithm can return a* $(k-\delta)$*-approximation to the distance between m pairs of nodes in a simple graph with m edges in time* $O(m^{1+\frac{1}{ck-d}-\varepsilon})$*. Consequently, there is no* $(k-\delta)$*-approximate distance oracle with* $O(m^{1+\frac{1}{ck-d}-\varepsilon})$ *preprocessing and* $O(m^{\frac{1}{ck-d}-\varepsilon})$ *query time.*

Finally, we prove a hardness result for dynamic approximate All Pairs Shortest Paths.

Theorem 2.5.13. *For every* $\varepsilon, \delta' > 0$ *and integer* $k \geqslant 4$ *there is a* $\delta > 0$ *such that if there is a dynamic algorithm for* $(k/2-\delta')$*-approximate APSP with preprocessing time* $O(N^3)$ *and update/query time* $O(m^{\frac{3-\omega}{2(k+1-\omega)}-\varepsilon})$ *in N-node and m-edge graphs, then there is an* $O(n^{2-\delta})$*-time algorithm for n-node graphs with maximum degree at most* $\sqrt{n}$ *that with probability at least* $9/10$ *answers for every edge whether it participates in a triangle.*

Proof. First, we run the subquadratic-time algorithm from Theorem 2.1.5 with ε' and α to be chosen later. We show how to use a dynamic algorithm to check for each edge $\{b,c\} \in B \times C$ whether it participates in a triangle. (Checking the edges in $A \times B \cup A \times C$ is symmetric.) For each G_i, we remove the $B \times C$ edges, obtaining a graph G_i'. We let G_1' be the input graph to be preprocessed by the dynamic algorithm in $O((n^{(1/2+\alpha)})^3) = O(n^{3/2+3\alpha})$ time, and we consider the following sequence of updates and queries we feed into the dynamic algorithm.

Sequence of updates and queries: For $1 \leqslant i \leqslant n^{3/2-3\alpha}$ phases, in each phase i we make the following queries and updates. Queries: For each edge $\{b,c\} \in (B \times C) \cap E(G_i)$, we query the pair $\{b,c\}$. This takes $O(n^{1/2+2\alpha})$ queries. Updates: we delete all the edges in G_i' and add all the edges in G_{i+1}', by using $O(n^{1/2+2\alpha})$ updates.

Postprocessing: We use the distance estimations returned by the queries to find for each edge $\{b,c\}$ in each G_i whether it in a triangle in G_i, as follows. For each G_i', we collect all the pairs $\{b,c\}$ for which the answer to the query is $< k$, and we refer to the corresponding edges as the candidates of G_i. For each candidate edge, we check whether it is in a triangle in G_i by iterating over all neighbors of the endpoints, and if so, we remove the edge from the set of candidates of G_j for every $j > i$. This finishes the reduction.

Running time: In total, for all phases, the number of queries and updates is $O(n^{2-\alpha})$. Hence, if each update and query takes time $O(n^{x(1/2+2\alpha)})$, we have that the total query and update time is $O(n^{x(1/2+2\alpha)} \cdot n^{2-\alpha})$, for all updates and queries. For $x < \frac{\alpha}{1/2+2\alpha}$, this is subquadratic.

For the postprocessing the analysis is similar to the one in Theorem 2.5.11: Checking whether a candidate edge forms a triangle takes time $O(n^{\alpha})$, and there are $O(n^{3/2+(\omega-1)/4+\varepsilon'+\alpha(k-3)})$ candidate edges in total, so the postprocessing takes total time $O(n^{3/2+(\omega-1)/4+\varepsilon'+\alpha(k-2)})$. For $\alpha = \frac{3-\omega}{4(k-2)} - \varepsilon''$, for $\varepsilon'' > \varepsilon'$, this is subquadratic. Since $\omega \geqslant 2$ and $k \geqslant 4$, this choice of α satisfies $\alpha < 1/6$, so also the preprocessing time of $O(n^{3/2+3\alpha})$ is subquadratic.

Furthermore, by a similar calculation to the one provided in Theorem 2.5.11, we have that $\frac{\alpha}{1/2+2\alpha} > \frac{3-\omega}{2(k+1-\omega)} - 2\varepsilon''$. For $\varepsilon > 2\varepsilon''$, we set $x = \frac{3-\omega}{2(k+1-\omega)} - \varepsilon < \frac{\alpha}{1/2+2\alpha}$. Hence, since the number of edges at any time of the dynamic process is $m = O(n^{1/2+2\alpha})$, as a function of the number of edges m, if the update time is $O(m^x) = O(m^{\frac{3-\omega}{2(k+1-\omega)}-\varepsilon})$, the total running time of the above algorithm is subquadratic, as desired. □

The following corollary follows immediately by combining Theorem 2.5.11, Theorem 2.5.13, and Theorem 2.4.1, where the first bullet follows by a straightforward reduction from the Offline Distance Oracles problem to Decremental Dynamic APSP (just preprocess the graph and answer the queries without ever making edge deletions).

Corollary 2.5.14 (Hardness of Approximation for Dynamic APSP)**.** *Let $k \geqslant 4$ be an integer, and let $\varepsilon, \delta > 0$, $c = \frac{4}{3-\omega}$ and $d = \frac{2\omega-2}{3-\omega}$. Assuming either the 3-SUM Conjecture or the APSP Conjecture:*

- *No algorithm can maintain a simple graph through a sequence of edge-deletion updates in a total of $O(m^{1+\frac{1}{ck-d}-\varepsilon})$ time, while answering distance queries between a given pair of nodes with a $(k-\delta)$-approximation in $O(m^{\frac{1}{ck-d}-\varepsilon})$ time.*
- *No algorithm can preprocess a simple graph in $O(n^3)$ time and then support (fully dynamic) updates and queries in $O(m^{\frac{1}{ck-d}-\varepsilon})$ time, where an answer to a query is a $(k-\delta)$-approximation to the distance between a given pair of nodes.*

2.6 Removing All 4-Cycles

In this section we prove Theorem 2.1.9 (Section 2.6.1), as well as some hardness consequences of it (Section 2.6.2).

2.6.1 A proof of Theorem 2.1.9

Theorem 2.1.9. *For any choice of constant $\alpha \in (0, \frac{3-\omega}{8})$ and $\varepsilon \in (0, \frac{3-\omega}{4} - 2\alpha)$ the following holds. Given a graph G with n vertices and maximum degree at most $\sqrt{n}$, there is a randomized algorithm, running in time $O(n^{2-\varepsilon})$, that returns a subset of the edges $E' \subseteq E(G)$ and a collection of $s = n^{3/2-3\alpha}$ subgraphs $G_1, \ldots, G_s \subseteq G$ such that:*

- *Every edge $e \in E'$ participates in a triangle of G.*
- *If an edge $e \in E(G)$ participates in a triangle of G, then it is either in E' or it participates in a triangle in at least one subgraph G_i.*
- *With high probability, each G_i has $O(n^{1/2+\alpha})$ vertices and maximum degree $O(n^{\alpha})$.*
- *With probability larger than 0.99, no subgraph G_i contains a 4-cycle.*

Proof. Let $\alpha \in (0, \frac{3-\omega}{8})$ and $\varepsilon, \varepsilon' \in (0, \frac{3-\omega}{4} - 2\alpha)$ to be chosen later. First, we run the subquadratic-time algorithm from Theorem 2.1.5 with ε', α, and $k = 4$. Recall that this algorithm returns a set of edges E' and $n^{3/2-3\alpha}$ subgraphs, such that each edge that is in a triangle is either in E' or in a triangle in one of the subgraphs, where each subgraph is a slice (A_j, B_k, C_ℓ), for $j, k, \ell \in [n^{1/2-\alpha}]$. Furthermore, each slice has $O(n^{(\omega-1)/4+\varepsilon'+4\alpha})$ 4-cycles in expectation, and therefore the overall number of 4-cycles in all the slices is at most $O(n^{3/2+\alpha+(\omega-1)/4+\varepsilon'})$ with probability at least $99/100$. Our algorithm adds more edges to E' such that the obtained slices are 4-cycle-free, as follows.

We show that it is possible to list all the 4-cycles in all the slices in time that is linear in their number, see Lemma 2.6.1 below. After listing all the 4-cycles, we denote by $\mathcal{S}_{j,k,\ell}$ the set of edges that participate in 4-cycles in the slice (A_j, B_k, C_ℓ). Note that after removing the edges $\mathcal{S}_{j,k,\ell}$ from the slice (A_j, B_k, C_ℓ) it becomes 4-cycle-free, as desired. It remains to check for each edge in $\mathcal{S}_{j,k,\ell}$ whether it participates in a triangle in the slice, and if so we add it to E'. Since the degree of each node in a slice is with high probability at most $O(n^{\alpha})$, this takes $O(|\mathcal{S}_{j,k,\ell}| \cdot n^{\alpha})$ time per slice with high probability. Hence, in total, for all slices, this takes time

$\sum_{j,k,\ell} O(|\mathcal{S}_{j,k,\ell}| \cdot n^{\alpha}) = O(n^{3/2+2\alpha+(\omega-1)/4+\varepsilon'})$ with constant probability. By setting $\alpha = (3-\omega)/8 - (\varepsilon+\varepsilon')/2$, this takes time $O(n^{2-\varepsilon})$, as desired. It remains to show how to efficiently list all the 4-cycles in all the slices in time that is linear in their number:

Lemma 2.6.1. *We can enumerate all 4-cycles in any of the slices (A_j, B_k, C_ℓ), for all $j,k,\ell \in [n^{1/2-\alpha}]$, in total time $O(n^{3/2+\alpha} + c)$ where c is the output size, that is, c is the total number of such 4-cycles.*

Proof. Observe that for any slice, any 4-cycle uses exactly two nodes from one of the sides of the slice. In the following, we show how to list all the 4-cycles that use two nodes from A_j, for all the slices (A_j, B_k, C_ℓ). Listing all the 4-cycles that use two nodes from B_k or two nodes from C_ℓ is symmetric.

We start with the following useful notations. For each slice (A_j, B_k, C_ℓ), we think about A_j as being the center of the slice, B_k being the left side of the slice, and C_ℓ being the right side of the slice. For a set A_j and a pair $\{a,a'\} \subseteq A_j$, let $L_k^{a,a'}$ be the set of nodes $b \in B_k$ for which $\{a,b,a'\}$ is a two-edge path. Similarly, let $R_\ell^{a,a'}$ be the set of nodes $c \in C_\ell$ for which $\{a,c,a'\}$ is a 2-path. Furthermore, for a set A_j and a pair $\{a,a'\} \subseteq A_j$, let $I_L^{a,a'}$ be the set of coordinates $k \in [n^{1/2-\alpha}]$ for which $L_k^{a,a'}$ is not empty. Similarly, $I_R^{a,a'}$ is the set of coordinates $\ell \in [n^{1/2-\alpha}]$ for which $R_\ell^{a,a'}$ is not empty. Finally, for every pair of sets A_j, B_k, let $P_{j,k}^L$ be the set of pairs $\{a,a'\} \subseteq A_j$ for which $|L_k^{a,a'}| \geqslant 2$. Similarly, $P_{j,\ell}^R$ is the set of pairs $\{a,a'\} \subseteq A_j$ for which $|R_\ell^{a,a'}| \geqslant 2$.

Our algorithm has a preprocessing step that computes all the sets $I_L^{a,a'}, I_R^{a,a'}$, all the nonempty sets $L_k^{a,a'}, R_\ell^{a,a'}$, and all the sets $P_{j,k}^L, P_{j,\ell}^R$ (for every $j,k,\ell \in [n^{1/2-\alpha}]$ and every pair $\{a,a'\} \in A_j$). Then we show that given the sets $I_L^{a,a'}, I_R^{a,a'}, L_k^{a,a'}, R_\ell^{a,a'}$ we can list all the 4-cycles that use two nodes from A_j and one node from each of B_k and C_ℓ, and given the sets $P_{j,k}^L, P_{j,\ell}^R, L_k^{a,a'}, R_\ell^{a,a'}$ we can list all 4-cycles between every pair A_j, B_k, and every pair A_j, C_ℓ. The details follow.

Preprocessing step: Recall that we denote by $N(u)$ the set of neighbors of a node u in the original graph. For each pair A_j, B_k, we go over all the nodes $b \in B_k$, and for each such node, we go over all the pairs $\{a,a'\} \subseteq N(b) \cap A_j$, and we add k to $I_L^{a,a'}$ and b to $L_k^{a,a'}$. If $|L_k^{a,a'}| \geqslant 2$, then we also add $\{a,a'\}$ to $P_{j,k}^L$. Since the size of B_k is $O(n^{1/2+\alpha})$, and the maximum degree of a node $b \in B_k$ in A_j is $O(n^{\alpha})$, for a pair A_j, B_k, this takes time $O(n^{1/2+\alpha} \cdot n^{2\alpha}) = O(n^{1/2+3\alpha})$. Hence, in total, for all pairs A_j, B_k, this takes time $O(n^{1-2\alpha} \cdot n^{1/2+3\alpha}) = O(n^{3/2+\alpha})$. The sets $I_R^{a,a'}$, the nonempty sets $R_\ell^{a,a'}$, and the sets $P_{j,\ell}^R$ are computed symmetrically, for all j, ℓ, and $\{a, a'\} \subseteq A_j$.

Listing all 4-cycles between all pairs A_j, B_k and all pairs A_j, C_ℓ: We show how to list all 4-cycles between all pairs A_j, B_k. Listing all the 4-cycles between all pairs A_j, C_ℓ is symmetric. Observe that the total number of 4-cycles between all pairs A_j, B_k is

$$\sum_{j,k \in [n^{1/2-\alpha}]} \sum_{\{a,a'\} \in P_{j,k}^L} \binom{|L_j^{a,a'}|}{2}$$

To list them, we go over all pairs A_j, B_k, and for each such pair we list all tuples (a, b, a', b'), where $\{a, a'\} \in P_{j,k}^L$, and $\{b, b'\} \subseteq L_k^{a,a'}$. Since all the sets $P_{j,k}^L$ and $L_k^{a,a'}$ were already computed in the preprocessing step, this takes an amount of time which is linear in the number of 4-cycles. We list all the 4-cycles between all pairs A_j, C_ℓ in a similar way.

Listing all 4-cycles that use two nodes from A_j and one node from each of B_k, C_ℓ, for every j, k, ℓ: Observe that the number of such 4-cycles is

$$\sum_{j \in [n^{1/2-\alpha}]} \sum_{\{a,a'\} \subseteq A_j} \sum_{(k,\ell) \in I_L^{a,a'} \times I_R^{a,a'}} |L_k^{a,a'}| \cdot |R_\ell^{a,a'}| \tag{1}$$

Our goal is to list all these 4-cycles in an amount of time that is linear in their number. For this, we go over all sets A_j, and for each such set, we go over all pairs $\{a, a'\} \subseteq A_j$, and for each such pair, we go over all pairs $(k, \ell) \in I_L^{a,a'} \times I_R^{a,a'}$, and list all the tuples (a, b, a', c), where $b \in L_k^{a,a'}$ and $c \in R_\ell^{a,a'}$. The amount of time for this step is proportional to

$$\sum_{j \in [n^{1/2-\alpha}]} \sum_{\{a,a'\} \subseteq A_j} \left(1 + \sum_{(k,\ell) \in I_L^{a,a'} \times I_R^{a,a'}} |L_k^{a,a'}| \cdot |R_\ell^{a,a'}|\right).$$

This is because for the pairs $\{a, a'\}$ that don't contribute any 4-cycle to the sum (1) we spend constant time. For the other pairs, the amount of time we spend is proportional to the number of 4-cycles they participate in. Note that the summand 1 contributes $O(n^{1/2-\alpha} \cdot |A_j|^2) = O(n^{1/2-\alpha} \cdot (n^{1/2+\alpha})^2) = O(n^{3/2+\alpha})$ to the running time. The other summand is simply the total number of 4-cycles as in (1). We thus obtain total time $O(n^{3/2+\alpha} + c)$, as desired. □

This finishes the proof of Theorem 2.1.9. □

2.6.2 Consequences of Theorem 2.1.9

We start with a reduction from triangle detection to 4-cycle detection.

Theorem 2.6.2. *For every $\delta > 0$ there is a $\delta' > 0$ such that if there is an $O(m^{1+\frac{3-\omega}{2(5-\omega)}-\delta})$-time algorithm for 4-cycle detection, then there is an $O(n^{2-\delta'})$-time algorithm for triangle detection in n-node graphs with maximum degree at most $\sqrt{n}$.*

Proof. First, we run the subquadratic-time algorithm from Theorem 2.1.9 with an arbitrarily small constant $\varepsilon > 0$ and $\alpha < (3-\omega)/8 - \varepsilon/2$ to be chosen later. Since the algorithm already checked for each edge in E' whether it participates in a triangle, and since each triangle either uses an edge from E' or is in one of the G_i's, it remains to solve triangle detection in each G_i.

For this, we add a dummy node bc on each edge $\{b, c\} \in B \times C$, which converts any triangle $\{a, b, c\}$ to a 4-cycle $\{a, b, bc, c\}$. Furthermore, since none of the G_i's had a 4-cycle before adding the dummy nodes, the existence of a 4-cycle in G_i implies the existence of a triangle. Therefore, to solve triangle detection, it suffices to run a 4-cycle detection algorithm in all the obtained G_i's. Let $T(m) = m^x$ be the time complexity for 4-cycle detection in m-edge graphs. Since each G_i has $O(n^{1/2+2\alpha})$ edges with high probability, the total running time for all the G_i's is

$$O(n^{3/2-3\alpha} \cdot (n^{1/2+2\alpha})^x) = O(n^{\frac{3}{2}-3\alpha+x(1/2+2\alpha)})$$

For $x < 1 + \alpha/(1/2 + 2\alpha)$, this is subquadratic. Hence, by setting $\alpha = (3 - \omega)/8 - \varepsilon'$, for some $\varepsilon' > \varepsilon/2$, the running time from Theorem 2.1.5 is subquadratic, and

$$1 + \alpha/(1/2 + 2\alpha) = 1 + \frac{\frac{3-\omega}{8} - \varepsilon'}{1/2 + 2(\frac{3-\omega}{8} - \varepsilon')}$$

$$> 1 + \frac{3-\omega}{2(5-\omega)} - \frac{8\varepsilon'}{2(5-\omega)}$$

By setting $x = 1 + \frac{3-\omega}{2(5-\omega)} - \delta$, for $\delta > 8\varepsilon'$, the total running time for 4-cycle detection in all the G_i's is subquadratic, as desired. □

Corollary 2.1.10 follows immediately from Theorem 2.6.2. The second and third bullets follow by essentially the same proof as the one provided for Theorem 2.6.2. Instead of running a 4-Cycle detection algorithm, we run a triangle detection in 4-cycle free graphs for the second bullet, and a girth approximation algorithm for the third bullet.

Corollary 2.6.3 (Hardness for Triangle in 4-Cycle-Free Graphs)**.** *Assuming that triangle detection in graphs with maximum degree at most $\sqrt{n}$ requires $n^{2-o(1)}$ time, no algorithm can solve any of the following problems in $O(m^{1+\frac{3-\omega}{2(5-\omega)}-\varepsilon})$ time, for any $\varepsilon > 0$:*

- *Decide if an m-edge graph has a 4-cycle.*
- *Decide if an m-edge 4-cycle-free graph has a triangle.*
- *Compute a $(5/3 - \delta)$-approximation to the girth of an m-edge graph, for any $\delta > 0$.*

2.7 On the Hardness of Triangle

The conditional lower bounds in our work are based on the $n^{2-o(1)}$ time hardness of two versions of triangle finding in $\sqrt{n}$-degree graphs: The *all-edge* version of reporting for each of the $n^{1.5}$ edges whether it is in a triangle, and the more basic *detection* version of just deciding if there is any triangle in the graph. The former is already known to be hard under either the 3SUM or APSP conjectures [268], two of the most central conjectures in fine-grained complexity [267, 270], and therefore does not need further justification (see also [136] for equivalences to range reporting problems). The goal of this section is to discuss the latter assumption.

Abboud and Vassilevska Williams [12] introduced the following Triangle Conjecture and used it to prove hardness result for dynamic problems; the conjecture has also been used elsewhere, e.g. in databases [96].

Conjecture 2.7.1 (The Triangle Conjecture [12])**.** *Triangle detection requires $m^{4/3-o(1)}$ time, for some density regime $m = \Omega(n)$. In other words, there exists a constant $1 \leqslant \alpha \leqslant$*

2 such that for all $\varepsilon > 0$ there is no algorithm that given a graph with n nodes and $m = \Theta(n^{\alpha})$ edges detects whether it contains a triangle in $O(m^{4/3-\varepsilon})$ time.

They also considered a weaker form of the conjecture where only some $\Omega(m^{1+\delta})$ lower bound is assumed, and a stronger form with an $m^{\frac{2\omega}{\omega+1}-o(1)}$ lower bound even when $\omega > 2$. However, the above $m^{4/3-o(1)}$ is the more natural and popular hypothesis and it continues to hold even if $\omega = 2$.

While the conjecture does not specify the density for which $m^{4/3-o(1)}$ time is required, by a simple high-degree low-degree analysis, one can show that the hardest regime is $m = n^{1.5}$:

Observation 2.7.2. The Triangle Conjecture is equivalent to the hypothesis that Triangle detection requires $n^{2-o(1)}$ time in graphs with average degree $\Theta(\sqrt{n})$.

Proof. One direction is trivial: If Triangle detection requires $n^{2-o(1)}$ time in graphs with average degree $\Theta(\sqrt{n})$, then for the density regime $m = \Theta(n^{3/2})$ Triangle detection requires $m^{4/3-o(1)}$ time, so the Triangle Conjecture holds.

For the other direction, suppose that Triangle in graphs with N nodes and $\Theta(N^{1.5})$ edges can be solved in $O(N^{2-\varepsilon})$ time, for some $\varepsilon > 0$. Given a graph on n nodes and m edges as input to Triangle, let H be the set of nodes of degree $\geqslant m^{1/3-\delta}$, and let $L = V \setminus H$ be the nodes of degree at most $m^{1/3-\delta}$.

- To find a triangle that uses any node from L, iterate over all m edges $\{u, v\}$ and if one of the endpoints is in L, e.g. u, scan its neighborhood and for each $w \in N(u)$ check if u, v, w is a triangle. This takes $O(m \cdot m^{1/3-\delta})$ time.
- To find a triangle that only uses nodes from H consider the induced graph on these $N = m/m^{1/3-\delta} = m^{2/3+\delta}$ nodes. This graph has only $m = O(N^{1.5})$ edges. If the number of edges happens to be $o(N^{1.5})$ we can artificially turn it into $\Theta(N^{1.5})$ by simply adding a bipartite graph on N nodes and $N^{1.5}$ edges (this does not introduce any new triangles). Then, by assumption, we can find a triangle in this graph in time $O(N^{2-\varepsilon}) = O((m^{2/3+\delta})^{2-\varepsilon}) = O(m^{4/3+2\delta-2/3\varepsilon}) = O(m^{4/3-\delta})$ for $\delta < \varepsilon/10$.

In both cases we can solve Triangle detection in time $O(m^{4/3-\delta})$, which refutes the Triangle Conjecture. □

This does not quite prove an equivalence between our hardness assumption and the Triangle Conjecture because we do not know how to reduce the *average* degree $\leqslant \sqrt{n}$ case to the *maximum* degree $\sqrt{n}$ case.

Indeed, this issue arises also in the all-edge version where we do not know how to reduce the general $m = n^{1.5}$ case to the $\sqrt{n}$-degree case. There, we sidestepped this discussion by starting from other popular conjectures (3SUM and APSP) rather than from a hardness assumption about All-Edges-Triangle itself. Can we do the same here? Unfortunately, basing the Triangle Conjecture on other popular conjectures such as 3SUM and APSP is a major open question:

Open Question 2.7.3. Can we prove the Triangle Conjecture under other hardness assumptions such as 3SUM or APSP?

Ever since Pătraşcu's [231] 3SUM-hardness for the all-edge and listing versions of Triangle, it has been a pressing open question to prove the same for detection. APSP has been connected to Triangle detection in the work of Vassilevska Williams and Williams [271] but only in a restricted sense: the two problems are subcubic-equivalent for *combinatorial* algorithms in *dense* graphs. Extending such results to general algorithms or to sparse graphs is a well-known challenge.

As a side result of independent interest, we make progress towards this goal. We prove the first conditional lower bound for Triangle detection that is based on the hardness of a problem of a very similar flavor to 3SUM and APSP: the Zero-Triangle problem. Importantly, this hardness continues to hold under the restriction to $\sqrt{n}$-degree graphs, justifying our belief that this is the hard case for triangles.

Definition 2.7.4 (Zero-Triangle)**.** Given a tripartite graph $G = (A \times B \times C, E)$ with integral edge weights $w : E \to [-W, +W]$ decide if there is a triangle $(a, b, c) \in A \times B \times C$ with total weight $w(a, b) + w(b, c) + w(a, c) = 0$.

Ignoring subpolynomial improvements, there are only two algorithms for this problem. The first is a brute force over all triples and its running time is $O(|A| \cdot |B| \cdot |C|)$. In the symmetric setting where $|A| = |B| = |C| = n/3$ this is $O(n^3)$ and it is optimal under both 3-SUM and APSP conjectures, as long as $W = \Omega(n^3)$ [231, 272]. The second algorithm is faster when W is small enough: It applies the standard exponentiation trick (encoding w as 2^w) to reduce summation to multiplication and then uses fast matrix multiplication. In the symmetric setting the running time is $O(W \cdot n^{\omega})$ and otherwise it is a complicated expression that depends on the rectangular matrix multiplication exponent. Assuming $\omega = 2$, the upper bound simplifies to $(W \cdot N)^{1+o(1)}$ where $N = (|A| \cdot |C| + |A| \cdot |B| + |B| \cdot |C|)$ is an upper bound on the size of the graph. It is natural to conjecture that these bounds cannot be broken for Zero-Triangle.

Conjecture 2.7.5 (The Strong Zero-Triangle Conjecture)**.** *Solving Zero-Triangle requires* $(\min\{WN, |A| \cdot |B| \cdot |C|\})^{1-o(1)}$ *time, for any parameters* $W, |A|, |B|, |C| = n^{\Theta(1)}$ *and where* $N = (|A| \cdot |C| + |A| \cdot |B| + |B| \cdot |C|)$.

While this conjecture is not known to be implied by the 3-SUM and APSP conjectures (because the existing reductions change the ratio of weight W to number of nodes n) its plausibility has the same source. In fact, it is analogous to the stronger version of the APSP conjecture recently studied by Chan, Vassilevska, and Xu [106]. Notably, Zero-Triangle is a problem that is hard due to the weights and the addition operator and *not* due to the graph structure: the input graph may be assumed to be complete. Thus, we find it surprising that it explains the hardness of our purely structural subgraph detection problems; in particular it gives a *tight* lower bound for Triangle:

Theorem 2.7.6. *If Triangle detection in graphs with maximum degree* $\sqrt{n}$ *can be solved in* $O(n^{2-\varepsilon})$ *time, for some* $\varepsilon > 0$*, then Zero-Triangle with* $|A| = n, |B| = |C| = \sqrt{n}$ *and* $W = \sqrt{n}$ *can be solved in* $O(n^{2-\varepsilon})$ *time, and the Strong Zero Triangle Conjecture is false.*

Proof. Given an instance of Zero-Triangle with $|A| = n, |B| = |C| = \sqrt{n}$ and $W = \sqrt{n}$ we construct an unweighted graph as follows. Each node in $u \in B \cup C$ is copied $6W + 1$ times $u_{-3W}, \ldots, u_{3W}$ where u_i represents both the node u and the integer value i. A node $a \in A$ has a single copy in the new graph.

An edge of weight x from $a \in A$ to $b \in B$ becomes an edge from a to b_x. An edge of weight y from $a \in A$ to $c \in C$ becomes an edge from a to c_{-y}. On the other hand, an edge of weight z from $b \in B$ to $c \in C$ becomes a matching between the b_i and c_j nodes such that there is an edge between b_i and c_{i+z} for all $i \in [-2W, 2W]$.

A zero-triangle (a, b, c) with weights $w(a, b) = x, w(a, c) = y, w(b, c) = z$ becomes a triangle a, b_x, c_{-y}. The edges (a, b_x) and (a, c_{-y}) exist by definition, and the third edge exists because $-y = x + z$. By a reverse argument, any triangle in the new graph corresponds to a zero-triangle in the original graph. $\square$

The reduction is rather simple but we find the statement quite interesting. First, it bases the Triangle Conjecture (and our hardness for 4-Cycle) on a hardness assumption of a very different nature. Second, it makes a substantial step towards establishing the Triangle Conjecture under the more central 3-SUM or APSP Conjectures. And third, assuming $\omega = 2$, it pinpoints a challenge that one must resolve before making any further progress on Triangle, the lower bound is completely tight for all density regimes (due to Observation 2.7.2).

2.8 Reduction from Triangle or 4-Cycle to any k-Cycle

For completeness, we include a proof of the following statement. The components of this proof are considered folklore.

Theorem 2.8.1. *For any integer $k \geqslant 3$ one of the following is true:*

- *There is a reduction that given an m-edge tripartite graph G runs in $O(m)$ time and constructs a graph $G^\star$ such that the k-cycles in $G^\star$ are in 1-to-1 correspondence with the triangles in G.*
- *There is a reduction that given an m-edge graph G runs in $O(m)$ time and constructs a graph $G^\star$ such that the k-cycles in $G^\star$ are in 1-to-1 correspondence with the 4-cycles in G.*

Recall that in Triangle detection we can assume without loss of generality that the input graph is tripartite, so this condition makes no big difference.

By Theorem 2.8.1, if k-Cycle detection can be solved in time $O(m^\alpha)$, for some $\alpha \geqslant 1$, then either Triangle or 4-Cycle detection can also be solved in time $O(m^\alpha)$. Moreover, if after $O(m^\alpha)$ preprocessing we can enumerate k-cycles with $m^{o(1)}$ delay, then the same is true for enumerating either triangles or 4-cycles.

Lemma 2.8.2. *Let $r \mid k$ be two positive fixed integers such that r divides k. There is a reduction that given an m-edge graph G runs in $O(m)$ time and constructs a graph $G^\star$ such that the k-cycles in $G^\star$ are in 1-to-1 correspondence with the r-cycles in G.*

Proof. We can assume that $3 \leqslant r < k$, as otherwise the claim is straightforward. Let G be a graph with m edges, we construct the graph $G^\star$ by replacing each edge of G with a path of length $\frac{k}{r}$ (that is, each edge of G is subdivided by $\frac{k}{r} - 1$ vertices).

The number of edges in $G^\star$ is $\frac{k}{r}m = O(m)$, and constructing $G^\star$ from G takes $O(m)$ time. We prove the claim by showing that $G^\star$ contains a k-cycle if and only if G contains an r-cycle.

If G contains an r-cycle, then after the subdivision of its edges this r-cycle corresponds to a k-cycle in $G^\star$. On the other hand, any simple cycle in $G^\star$ can be partitioned into paths of length $\frac{k}{r}$ corresponding to the full subdivision of edges from G. This holds as the degree of every subdividing vertex is exactly 2. Hence, every cycle in $G^\star$ is of size divisible by $\frac{k}{r}$ and such cycle of size $\frac{k}{r} \cdot x$ must correspond to a cycle of size x in G. In particular, if $G^\star$ contains a k-cycle then G contains an r-cycle. □

Lemma 2.8.3. *Let $k \geqslant 3$ be any* ***odd*** *fixed integer. There is a reduction that given an m-edge tripartite graph G runs in $O(m)$ time and constructs a graph $G^\star$ such that the k-cycles in $G^\star$ are in 1-to-1 correspondence with the triangles in G.*

Proof. Let G be a tripartite graph with m edges and vertex sets $V = A \cup B \cup C$. We construct $G^\star$ by replacing every edge of G with endpoints in B and C with a path of length $k-2$ (that is, we subdivide each edge of $E(G) \cap (B \times C)$ by $k-3$ vertices). The number of edges in $G^\star$ and the time to construct it are $O(m)$. If G contains a triangle then $G^\star$ clearly contains a corresponding k-cycle. It is left to prove that if $G^\star$ contains a k-cycle then G contains a triangle.

Denote by $D^{(i)}$ for $1 \leqslant i \leqslant k-3$ the set of all i-th vertices in a subdivision of some subdivided edge. The graph $G^\star$ is homomorphic to the k-cycle by the partition $A, B, D^{(1)}, \ldots, D^{(k-3)}, C$. Any k-cycle in $G^\star$ must include exactly one vertex in each of the parts $A, B, D^{(1)}, \ldots, D^{(k-3)}, C$, since the k-cycle is not bipartite yet after the removal of any of these parts the remaining graph is homomorphic to a path and hence bipartite. Due to the degree of each vertex in a part $D^{(i)}$ being exactly 2, such a cycle is necessarily a triangle of G with one subdivided edge. This follows in a similar manner to the proof of Lemma 2.8.2. □

Proof of Theorem 2.8.1. Let $k \geqslant 3$. If k is not a power of 2, then it has an odd prime divisor p and hence we can apply Lemma 2.8.3 to reduce from Triangle detection to p-Cycle detection, and then apply Lemma 2.8.2 to reduce from p-Cycle detection to k-Cycle detection, to prove the theorem. Otherwise, $k \geqslant 3$ is a power of 2 and in particular is divisible by 4. Then we can use Lemma 2.8.2 to reduce from 4-Cycle detection to k-Cycle detection. □

We note that the components in the proof of Theorem 2.8.1 (and any other previously known technique) do not show that if 4-Cycle detection is linear then so is Triangle detection. The reason that a similar argument fails is that as a bipartite graph, a 4-cycle can appear between any of the three pairs of parts in G. On the other hand, we observe that if the original graph G contains no 4-cycle, then a similar reduction does work.

Chapter 3

Hardness of Approximation for Distributed Diameter

This chapter is based on a joint work with Ofer Grossman and Ami Paz [170], in which we show improved hardness of approximation results for diameter computation in the CONGEST model.

3.1 Introduction

We study the problem of computing the exact or approximate value of the diameter D in the distributed CONGEST model [7,100,101,155,178,179,181,184,207,236].

As discussed in Chapter 1, Section 1.2.3, computing the exact diameter takes $\Theta(n/\log n + D)$ rounds [155,184], whereas a $1/2$-approximation takes only $O(D)$ rounds by running a BFS from some node v and returning the depth of the BFS tree. Moreover, a simple indistinguishability argument shows that there is no $o(D)$-round algorithm for any constant approximation.

This leads to the following natural question: What is the best approximation value $1/2 < \alpha < 1$ for which there is a *time-optimal* distributed CONGEST algorithm that finds an α-approximation? In other words, what's the best approximation factor that we can find in $O(D)$ rounds?

Our main result in this work is that there is no algorithm that finds better than a $6/11$-approximation in $O(D)$ rounds. Specifically, we show that even for constant values of D, any such algorithm must take $\Omega(n^{1/6}/\log n)$ rounds.

Approx.	Bound	Ref. and Comments
Exact	$\tilde{\Theta}(n)$	[155, 184]
2 vs. 3	$\tilde{\Omega}(n)$	[181]
$\lfloor 2/3D \rfloor < \tilde{D} \leqslant D$	$O(n^{1/2} + D)$	[178]
$2/3 + \epsilon$	$\tilde{\Omega}(n)$	[7]
3 vs. 5	$\tilde{\Omega}(n)$	**This work** (Theorem 3.1.4)
$3/5 + \epsilon$	$\tilde{\Omega}(n^{1/3})$	**This work** (Theorem 3.1.3)
$4/7$	$O(n^{1/3} + D)$	[32]
$4/7 + \epsilon$	$\tilde{\Omega}(n^{1/4})$	**This work** (Theorem 3.1.2)
$6/11 + \epsilon$	$\tilde{\Omega}(n^{1/6})$	**This work** (Theorem 3.1.1)
$1/2$	$O(D)$	Folklore

Table 3.1: A summary of the state of the art results for diameter approximation.

3.1.1 Our Contribution

Our main result is the following theorem.

Theorem 3.1.1. *For any constant $0 < \epsilon < 5/11$, any algorithm for finding a $(6/11 + \epsilon)$-approximation for the diameter in the CONGEST model requires $\Omega(n^{1/6}/\log n)$ rounds.*

We prove analogous theorems for $(4/7 + \epsilon)$-approximation and $(3/5 + \epsilon)$-approximation, with lower bounds of $\Omega(n^{1/4}/\log n)$ and $\Omega(n^{1/3}/\log n)$, respectively.

Theorem 3.1.2. *For any constant $0 < \epsilon < 3/7$, any algorithm for finding a $(4/7 + \epsilon)$-approximation for the diameter in the CONGEST model requires $\Omega(n^{1/4}/\log n)$ rounds.*

Theorem 3.1.3. *For any constant $0 < \epsilon < 2/5$, any algorithm for finding a $(3/5 + \epsilon)$-approximation for the diameter in the CONGEST model requires $\Omega(n^{1/3}/\log n)$ rounds.*

These results hold even against constant diameter graphs and even against randomized algorithms that succeed with probability at least 2/3. Prior to our work, besides the near-linear lower bound of Frischknecht et al. [155] for exact diameter, only a lower bound for $(2/3 + \epsilon)$-approximation was known: In the same work by Frischknecht et al., they showed an $\Omega(\sqrt{n}/\log n)$ lower bound for this approximation factor, and Abboud et al. [7] improved their result all the way up to $\Omega(n/\log^3 n)$ lower bound. Theorems 3.1.1, 3.1.2, and 3.1.3, as well as the aforementioned lower bounds, also apply for algorithms that allow a constant *additive* error, in addition to the multiplicative one, as we explain in Section 3.1.2.

Diameter 3 vs 5: Next, we prove that distinguishing graphs of diameter 3 from graphs of diameter 5 requires a near-linear number of rounds.

Theorem 3.1.4. *Any algorithm for distinguishing graphs of diameter 3 from graphs of diameter 5 in the CONGEST model requires $\Omega(n/\log n)$ rounds.*

We find this result rather surprising. There exists an algorithm [178] running in $O(\sqrt{n\log n}+D)$ rounds and returning an estimate $\lfloor\frac{2D}{3}\rfloor \leqslant \tilde{D} \leqslant D$. While the rounding in this equation might seem like an artifact of the proof, Theorem 3.1.4 shows that it is actually necessary.

That is, an algorithm for finding an estimate $\frac{2}{3}D \leqslant \tilde{D} \leqslant D$ can be used to distinguish diameter 3 from diameter 5, and we show that such a distinction must require $\Omega(n/\log n)$ rounds — much more than the $O(\sqrt{n\log n}+D)$ running time of the algorithm of [178].

Our proofs use the well-established technique of reductions from Communication Complexity to the CONGEST model [7, 43, 99, 103, 119, 132, 140, 150, 155, 162, 168, 180, 224, 237, 252]. Our main technical novelty is an interesting connection between extremal combinatorics, and specifically the existence of *generalized polygons* [156], and diameter approximation. This extends prior work connecting extremal combinatorics and distributed computing [102, 132, 150].

3.1.2 Robust Approximation

When dealing with diameter approximation, an important distinction to make is between *robust* and *non-robust* lower bounds. For example, as discussed above, an algorithm that finds an approximation $\tilde{D}$ of the diameter satisfying $\lfloor\frac{2D}{3}\rfloor \leqslant \tilde{D} \leqslant D$ does not in general imply a $\frac{2}{3}$-approximation.

However, as the diameter gets larger, the approximation ratio does approach $2/3$. One way to view this is by saying that our 3 vs 5 lower bound is not a "robust" lower bound for $(3/5+\epsilon)$-approximation. To show a "robust" lower bound for $(3/5+\epsilon)$-approximation, we need a stronger result, i.e., that for any constant β, it is hard to distinguish between graphs of diameter $(3/5)D-\beta$ and graphs of diameter D. This would show that finding a $(3/5+\epsilon)$-approximation of the diameter is hard not only in some low-diameter graphs, but also more generally. We formally define the notions of α-approximation for diameter and robust diameter lower bound.

Definition 3.1.5 (α-approximation for diameter)**.** We say that an estimate $\tilde{D}$ is an α-approximation for diameter if

$$\alpha D \leqslant \tilde{D} \leqslant D.$$

Definition 3.1.6 (Robust diameter lower bound)**.** We say that α-approximating the diameter is *robustly* $T(n)$-hard if for any constant β, there is no algorithm which returns a value $\tilde{D}$ satisfying

$$\alpha D - \beta \leqslant \tilde{D} \leqslant D$$

in $o(T(n))$ rounds in the CONGEST model.

In this work, we prove both robust and non-robust lower bounds. The lower bounds presented in Theorems 3.1.1, 3.1.2, and 3.1.3 are robust, while the lower bound that is presented in Theorem 3.1.4 is not robust.

Notably, for a $(3/5+\epsilon)$-approximation, we give a robust $\Omega(n^{1/3}/\log n)$ lower bound, and a non-robust lower bound of $\Omega(n/\log n)$. The work of [178] rules out a robust lower bound better than $\Omega(\sqrt{n \log n})$, even for 2/3-approximation. This shows that there is an inherent, and large, gap between the robust and non-robust lower bounds.

This distinction between robust and non-robust approximation has been noted before, though not using this terminology. Holzer and Wattenhofer [181] showed that distinguishing diameter 2 from diameter 3 requires $\Omega(n/\log n)$ rounds, a result that can be viewed as a non-robust $(2/3+\epsilon)$-approximation lower bound. As discussed earlier, a robust lower bound of $\Omega(n/\log^3 n)$ for the same approximation factor was later proven by Abboud et al. [7].

3.1.3 Further Related Work

The lower bound of Abboud et al. [7] for $(2/3+\epsilon)$-approximation follows from a lower bound for distinguishing between diameter $4\ell+2$ and $6\ell+1$, for some constant $\ell > 1$. Bringmann and Forster improved this result by showing the same hardness for distinguishing diameter $2\ell+1$ and $3\ell+1$ [90].

In a concurrent work [32], the authors show an upper bound of $O(n^{1/3}+D)$ for computing a $4/7$-approximation for diameter.

All the results that are presented in this work are for unweighted graphs. For weighted graphs, Holzer and Pinsker [179] showed that $(1/2+\epsilon)$-approximation requires $\Omega(n/\log n)$ rounds. For $(1/2)$-approximation in the weighted case, one can compute single source shortest paths. The state of the art algorithm for single source shortest paths in the CONGEST model is by Forster and Nanongkai [154], who showed two algorithms for the problem. The first running in $\tilde{O}(\sqrt{nD})$ rounds and the second running in $\tilde{O}(\sqrt{n}D^{1/4}+n^{3/5}+D)$ rounds.

Road-map: In Section 3.2 we start with some basic definitions. The technical heart of this chapter is in Sections 3.3 and 3.4. Theorem 3.1.4 is proved in Section 3.3, and Theorems 3.1.1, 3.1.2, and 3.1.3 are proved in Section 3.4.

3.2 Preliminaries

3.2.1 Basic Notations

For a graph H that is not the input graph, we denote its set of nodes and edges by V_H and E_H, respectively. The distance between two nodes u,v in a graph G is denoted by $d_G(u,v)$, and is the minimum number of hops in a path between them in G. The diameter D of the graph is the maximum distance between two nodes in it. The girth of the graph g is the minimum length of a cycle in it.

3.2.2 Communication Complexity

In the two-party communication setting [205,274], two players, Alice and Bob, are given two input strings, $x,y \in \{0,1\}^K$, respectively, and need to jointly compute a function $f : \{0,1\}^K \times \{0,1\}^K \to \{\text{TRUE}, \text{FALSE}\}$ of their inputs, using a predefined communication protocol. The *communication complexity* of a function f is defined as follows. Definition 3.2.1 is a special case of Definition 6.2.1 that will be used later in Chapter 6.

Definition 3.2.1. [Communication Complexity]
Let $K \geqslant 1$ be an integer, f be a function $f : \{0,1\}^K \times \{0,1\}^K \to \{\text{TRUE}, \text{FALSE}\}$, and $\mathcal{Q}$ be the family of protocols that compute f correctly with probability at least

$2/3$. Given 2 inputs $x, y \in \{0,1\}^K$, denote by $\pi_Q(x,y)$ the transcript of a protocol Q on the inputs x, y, i.e., the sequence of bits that are exchanged between Alice and Bob. The cost of a protocol Q is $\text{Cost}(Q) = \max_{x,y\in\{0,1\}^k} |\pi_Q(x,y)|$.

The communication complexity of f, denoted by $CC_f(K)$, is defined to be the minimum cost over all the possible protocols that compute f correctly with probability at least $2/3$: $CC_f(K) = \min_{Q\in\mathcal{Q}} \text{Cost}(Q)$.

Definition 3.2.2. [Set-Disjointness] The Set-Disjointness function is defined as follows. For two strings $x, y \in \{0,1\}^K$, we say that x and y are not disjoint if and only if there is some index $i \in [K]$ such that $x_i = y_i = 1$. Otherwise we say that the strings are disjoint.

It is well known that the communication complexity of Set-Disjointness is $\Omega(K)$ [254].

Remark 3.2.3. Adding 0 bits to both input strings in matching locations does not change the output. Thus, we can assume a constant fraction of both input strings is 0 without affecting the asymptotic communication complexity. We use this fact in Section 3.4.

3.2.3 Lower Bound Graphs

Our lower bounds use the standard notion of family of lower bound graphs (see, e.g., [103]).

Definition 3.2.4. (Family of Lower Bound Graphs)
Let $K > 1$ be an integer, $f : \{0,1\}^K \times \{0,1\}^K \to \{\text{TRUE}, \text{FALSE}\}$, and P be a graph predicate. A family of graphs $\{G_{(x,y)} = (V_{(x,y)}, E_{(x,y)}) \mid x, y \in \{0,1\}^K\}$ where each $G_{(x,y)}$ has a partition of the set of nodes $V_{(x,y)} = V_A \dot{\cup} V_B$ is said to be a family of *lower bound graphs w.r.t. f and P* if the following properties hold:

1. Only the existence of nodes in V_A or edges in $V_A \times V_A$ may depend on x;
2. Only the existence of nodes in V_B or edges in $V_B \times V_B$ may depend on y;
3. $G_{(x,y)}$ satisfies the predicate P iff $f(x,y) = \text{TRUE}$.

For such a family, we denote by $cut(A,B)$ the the set of edges between V_A and V_B. We use the following theorem, which is standard in the context of reductions to communication complexity (see, for example [7,103,132,155,179]). Its proof is by a standard simulation argument and appears in [103].

Theorem 3.2.5. *Fix a function $f : \{0,1\}^K \times \{0,1\}^K \to \{\text{TRUE}, \text{FALSE}\}$ and a predicate P. If there is a family of lower bound graphs w.r.t. f and P, then any algorithm for deciding P in the CONGEST model requires at least $\Omega\left(\frac{CC_f(K)}{|cut(A,B)|\log n}\right)$ rounds.*

d

3.2.4 Generalized Polygons

Our proofs in Section 3.4 use the existence of generalized polygons [156]. A generalized polygon is an incidence relation whose incidence graph has several nice properties. In our context, we use the following key property of a a generalized polygon's incidence graph: its girth is twice its diameter. For the sake of simplifying the presentation, we also use the fact that the incidence graphs are balanced.

We use the notation $H = (L, R, E_H)$ to denote a bipartite graph H, where the bi-partition of the vertex set of H is L and R, and the set of edges of H is E_H. When $|L| = |R| = p$, we say that H is a *balanced* bipartite graph of size $2p$.

Definition 3.2.6. For two integers $p \geqslant t \geqslant 3$, we denote by $\mathrm{Ex}(p,t)$ the maximum number of edges in a balanced bipartite graph of size $2p$, diameter t, and girth $2t$.

For $t \in \{3,4,6\}$, there are generalized polygons whose incidence graph has $2p$ nodes, diameter t, girth $2t$, and $\Theta(p^{1+\frac{1}{t-1}})$ edges. The cases of $t = 3$ and $t = 4$ were shown by Singelton [259], and Benson [72] gave a simplified proof and extended the result for $t = 6$. This is summarized in the following theorem, which will be used later without explicitly re-mentioning generalized polygons.

Theorem 3.2.7 ([72,259]). *For $t \in \{3,4,6\}$, it holds that $Ex(p,t) = \Omega(p^{1+\frac{1}{t-1}})$.*

3.3 Diameter 3 vs 5

In this section we prove the following theorem.

Theorem 3.1.4 *Any algorithm for distinguishing graphs of diameter 3 from graphs of diameter 5 in the CONGEST model requires $\Omega(n/\log n)$ rounds.*

To prove Theorem 3.1.4, we show a family of lower bound graphs $\{G_{(x,y)} \mid x,y \in \{0,1\}^K\}$ with respect to the Set-Disjointness function and the graph predicate that distinguishes between graphs of diameter 3 and graphs of diameter 5. That is, the predicate is defined only on a graph G with either 3 or 5, and is TRUE if and only if G has diameter 5. We start with the fixed graph construction G and then we show how to get the graph $G_{(x,y)}$ given two strings $x,y \in \{0,1\}^K$.

The Fixed Graph Construction G: The fixed graph construction is defined as follows. There are 8 sets of nodes $S, C^1, A^1, B^1, C^2, A^2, B^2, T$, each of size $p = n/8$. Each of the sets S, A^1, B^1, A^2, B^2, T is an independent set, and C^1 and C^2 are cliques. The nodes in the sets are denoted $S = \{s_i \mid i \in [p]\}$, $T = \{t_i \mid i \in [p]\}$, and for $h \in \{1,2\}$, $A^h = \{a_i^h \mid i \in [p]\}$, $B^h = \{b_i^h \mid i \in [p]\}$, and $C^h = \{C_i^h \mid i \in [p]\}$.

The connections between the sets are defined as follows. Each pair of sets $H_1 \neq H_2 \in \{S, C^1, A^1, B^1\}$ is connected by a perfect matching, where we connect the i'th node in H_1 to the i'th node in H_2. For example, the sets S and C^1 are connected by the perfect matching $\{(s_i, c_i^1) \mid i \in [p]\}$. Similarly, each pair of sets in $\{T, C^2, A^2, B^2\}$ is connected by a perfect matching. This concludes the fixed graph construction G. Let $K = p^2$. We define the graph $G_{(x,y)}$, given two strings $x,y \in \{0,1\}^K$, as follows.

Obtaining $G_{(x,y)}$ from G and $x,y \in \{0,1\}^K$: For each of the strings x and y, we index the $K = p^2$ positions by $x_{(i,j)}$ and $y_{(i,j)}$ for $i,j \in [p]$. The set of nodes of $G_{(x,y)}$ is exactly as in G. The set of edges of $G_{(x,y)}$ contains all the edges in G, and the following edges between pairs of nodes in $A^1 \times A^2$ and between pairs of nodes in $B^1 \times B^2$.

$$\{(a_i^1, a_j^2) \mid x_{(i,j)} = 0\}; \quad \{(b_i^1, b_j^2) \mid y_{(i,j)} = 0\}.$$

That is, if $x_{(i,j)} = 0$, we add an edge between a_i^1 and a_j^2, and if $y_{(i,j)} = 0$, we add an edge between b_i^1 and b_j^2. This concludes the definition of $G_{(x,y)}$ (See also Figure 3.1, for an illustration). Next, we prove that $G_{(x,y)}$ has diameter 3 if the strings x and y are disjoint, and otherwise it has diameter at least 5. We prove this in Lemmas 3.3.1 and 3.3.2.

Lemma 3.3.1. *If the strings x and y are disjoint, then the diameter of $G_{(x,y)}$ is* 3*.*

Proof. We show that for any two nodes u, v, $d_{G_{(x,y)}}(u,v) \leqslant 3$. Let $L = S \cup C^1 \cup A^1 \cup B^1$, and let $R = T \cup C^2 \cup A^2 \cup B^2$ The proof is by the following case analysis.

1. $u, v \in L$ or $u, v \in R$: We prove the claim for the case in which $u, v \in L$. The case for which $u, v \in R$ is similar. Observe that any node in L is connected by an edge to some node in C^1. Hence, since C^1 is a clique, this implies that $d_{G_{(x,y)}}(u,v) \leqslant 3$.

2. $u \in L$ and $v \in R$: Hence, u belongs to one of the sets in $\{S, C^1, A^1, B^1\}$ and v belongs to one of the sets in $\{T, C^2, A^2, B^2\}$. We assume that $u \in S$ and $v \in T$; the proof for the other cases is similar. Let i be such that $u = s_i$, and j such that $v = t_j$. Since the sets are disjoint, it holds that either $x_{(i,j)} = 0$, or $y_{(i,j)} = 0$ (or both). Hence, either there is an edge between a^1_i and a^2_j, or there is an edge between b^1_i and b^2_j (or both), and assume the former without loss of generality. Since s_i is connected to a^1_i and t_j is connected to a^2_j, we have $d_{G_{(x,y)}}(s_i, t_j) \leqslant 3$. Furthermore, one can verify that the distance between s_i and t_j cannot be smaller than 3, which implies that the diameter of the graph is 3. □

Lemma 3.3.2. *If the strings x and y are not disjoint, then the diameter of $G_{(x,y)}$ is at least 5.*

Proof. As the sets are not disjoint, there are $i, j \in [p]$ for which it holds that $x_{(i,j)} = y_{(i,j)} = 1$. We show that in this case, any path P from s_i to t_j is of length at least 5, i.e., $d_{G_{(s_i,t_j)}}(u,v) \geqslant 5$. Observe that any path P from s_i to t_j must either pass from a node in A^1 to a node in A^2, or from a node in B^1 to a node in B^2. We assume that former case; the latter is similar. The proof is by the following case analysis.

1. The path P visits a node $a^2_{j'} \in A^2$ for which $j' \neq j$: Observe that $d_{G_{(x,y)}}(s_i, a^2_{j'}) \geqslant 2$, and that $d_{G_{(x,y)}}(a^2_{j'}, t_j) = 3$. Hence, $d_{G_{(x,y)}}(s_i, t_j) \geqslant 5$.

2. The path P visits a^2_j. Since $x_{(i,j)} = 1$, there is no edge between a^1_i and a^2_j. This implies that $d_{G_{(x,y)}}(s_i, a^2_j) \geqslant 4$, and hence $d_{G_{(x,y)}}(s_i, t_j) \geqslant 5$. □

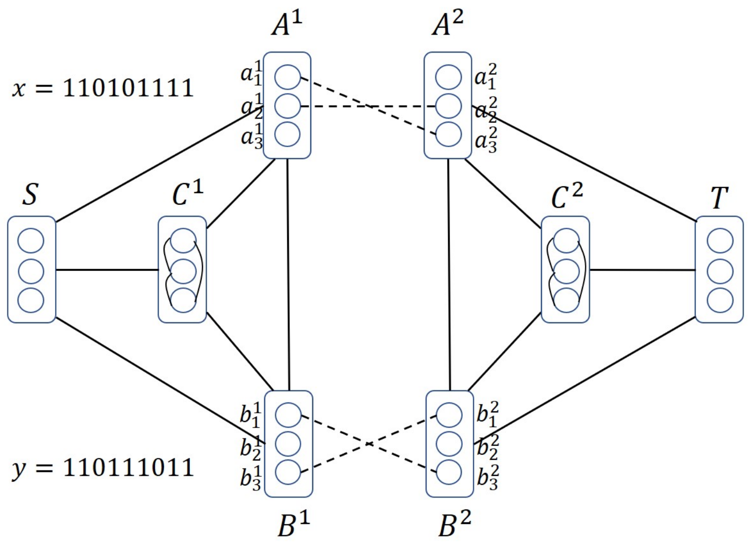

Figure 3.1: Diameter 3 vs 5: An example for the graph construction $G_{(x,y)}$ for $p = 3$: There are 8 sets of nodes $S, C^1, A^1, A^2, C^2, B^1, B^2, T$, each of size $p = 3$. Each of the sets S, A^1, A^2, B^1, B^2, T forms an independent set, and the sets C^1 and C^2 are cliques. In this diagram, an edge between two sets represents a perfect matching connecting them. For example, the edge between S and C^1 represents all the edges in $\{(s_i, c_i^1) \mid i \in [p]\}$. The dashed edges between A^1 and A^2 are the input edges which depend on the input sting x. Recall that we index the $p^2 = 9$ positions of x by pairs of indices $(i, j) \in [p] \times [p]$. In this example, we have that $x_{(1,3)} = x_{(2,2)} = 0$, and all the other bits of x are 1's. Hence, the only edges between A^1 and A^2 are (a_1^1, a_3^2) and (a_2^1, a_2^2). Similarly, the dashed edges between B^1 and B^2 represent the input edges which depend on the string y. Since in this example we have $y_{(1,3)} = y_{(3,1)} = 0$, and all the other bits of y are 1's, the only edges between B^1 and B^2 are (b_1^1, b_3^2) and (b_3^1, b_1^2).

Proof of Theorem 3.1.4. First, we define $V_A = S \cup T \cup A^1 \cup A^2 \cup C^1 \cup C^2$, and $V_B = B^1 \cup B^2$. Lemmas 3.3.1 and 3.3.2 imply that $\{G_{(x,y)} \mid x, y \in \{0,1\}^K\}$ is a family of lower bound graphs with respect to the Set-Disjointness problem and the graph predicate that distinguishes between graphs of diameter 3 and graphs of diameter 5. Observe that the cut size is $E(V_A, V_B) = \Theta(p)$, and $p = \Theta(n)$. Hence, since the length of the input strings is $K = p^2$, and since the communication complexity of

Set-Disjointness is $\Omega(K) = \Omega(p^2)$, Theorem 3.2.5, implies that any algorithm for deciding whether a graph has diameter 3 or 5 in the CONGEST model requires $\Omega(p^2/p\log p) = \Omega(p/\log p) = \Omega(n/\log n)$ rounds. □

The connectivity of $G_{(x,y)}$: One may wonder about the connectivity of $G_{(x,y)}$. If the graph $G_{(x,y)}$ is not connected, then the construction wouldn't be meaningful as there is a trivial lower bound of $\Omega(D)$, where D is the diameter of the graph, which is ∞ in graphs that are not connected. Observe that the only case in which $G_{(x,y)}$ is not connected is when $x = y = 1^K$. To ensure connectivity (and in fact constant diameter, due to the cliques C^1 and C^2), we can assume that at least one of the strings x or y has a zero bit. Clearly, the communication complexity of Set-Disjointness doesn't change under this assumption. In fact, Remark 3.2.3 allows to make an even stronger assumption, which we only need in the next section.

3.4 Robust Lower Bounds

In this section we prove robust lower bounds for $(6/11+\epsilon)$-approximation, $(4/7+\epsilon)$-approximation, and $(3/5+\epsilon)$-approximation of the diameter.

Theorem 3.4.1. *Let $t \in \{3,4,6\}$. For any constant $0 < \epsilon < 1 - \frac{t}{2t-1}$, any algorithm for computing a $(\frac{t}{2t-1} + \epsilon)$-approximation to the diameter in the CONGEST model requires $\Omega\left(n^{1/t}/\log n\right)$ rounds, where n is the number of nodes in the input graph.*

The theorem has the following consequences, when plugging in $t = 6$, $t = 4$ and $t = 3$, in this order.

Theorem 3.1.1 *For any constant $0 < \epsilon < 5/11$, any algorithm for finding a $(6/11 + \epsilon)$-approximation for the diameter in the CONGEST model requires $\Omega(n^{1/6}/\log n)$ rounds.*

Theorem 3.1.2 *For any constant $0 < \epsilon < 3/7$, any algorithm for finding a $(4/7+\epsilon)$-approximation for the diameter in the CONGEST model requires $\Omega(n^{1/4}/\log n)$ rounds.*

Theorem 3.1.3 *For any constant $0 < \epsilon < 2/5$, any algorithm for finding a $(3/5+\epsilon)$-approximation for the diameter in the CONGEST model requires $\Omega(n^{1/3}/\log n)$ rounds.*

Recall that $\text{Ex}(p,t)$ is the maximum number of edges of a balanced bipartite graph of size $2p$, diameter t, and girth $2t$ (see Definition 3.2.6). Let $t \in \{3,4,6\}$, $p \geqslant$

t and let $K = \mathrm{Ex}(p,t)$. To prove Theorem 3.4.1, we show a family of lower bound graphs $\{ G_{(x,y)} \mid x,y \in \{0,1\}^K \}$ with respect to the Set-Disjointness function and the graph predicate that distinguishes between graphs of diameter $t(b+1)+1$ and graphs of diameter $(2t-1)b$, for some integer $b = \Theta(1/\epsilon)$ that will be chosen later.

The rest of this section is organized as follows. In section 3.4.1, we start with the description of $G_{(x,y)}$ given two strings $x,y \in \{0,1\}^K$. In section 3.4.2, we show that $\{ G_{(x,y)} \mid x,y \in \{0,1\}^K \}$ is a family of lower bound graphs with the required properties. In Section 3.4.3, we deduce Theorem 3.4.1. While the graphs $G_{(x,y)}$ need to be connected, we ignore this fact in Section 3.4.1; in Section 3.4.4 we show how to slightly modify the construction so that the graphs become connected (and even of constant diameter) for any x and y.

3.4.1 Description of $G_{(x,y)}$

Given two strings $x,y \in \{0,1\}^K$, we describe the graph $G_{(x,y)}$ in three steps. In the first step, given a string $z \in \{0,1\}^K$, we define a bipartite graph H^z. Roughly speaking, H^z is obtained from a densest possible balanced bipartite graph H of size $2p$, diameter t and girth $2t$, where we keep only some of the edges of H in H^z according to the string z. In the second step, we define a graph $\widetilde{H}^z$, which is obtained form H^z by stretching each edge to a path of length b. In the third step, we describe how to get $G_{(x,y)}$ from $\widetilde{H}^x$ and $\widetilde{H}^y$.

Description of H^z: Let $H = (L,R,E_H)$ be a balanced bipartite graph of size $2p$, diameter t, girth $2t$, and a maximum number of edges. That is, the number of edges of H is $|E_H| = \mathrm{Ex}(p,t) = K$. We denote the nodes of H by $L = \{\ell_1,\cdots,\ell_p\}$ and $R = \{r_1,\cdots,r_p\}$. Furthermore, let $\pi : E_H \to [K]$ be an enumeration of E_H, that is, π is an arbitrary ordering over the set of pairs $E_H \subseteq L \times R$. By this mapping, each bit of a string $z \in \{0,1\}^K$ corresponds to a unique edge in E_H.

Given a string $z \in \{0,1\}^K$, the graph H^z is defined as follows. H^z is a version of H where we keep only the edges for which the corresponding bits in z are 0. More formally, $H^z = (L^z, R^z, E_{H^z})$ is a balanced bipartite graph with $|L^z| = |R^z| = p$, where $L^z = \{\ell^z_1,\cdots,\ell^z_p\}$ and $R^z = \{r^z_1,\cdots,r^z_p\}$. A pair of nodes $(\ell^z_i, r^z_j) \in L^z \times R^z$ is connected by an edge in H^z if (ℓ_i, r_i) is an edge of H and $z_{\pi(\ell_i,r_i)} = 0$, that is,

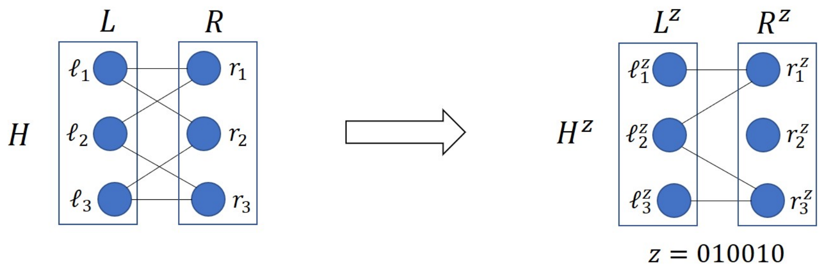

Figure 3.2: An example for H and H^z. In this example $t = p = 3$ and therefore H is a bipartite graph of diameter $t = 3$ and girth $2t = 6$. For these parameters, we have $K = \text{Ex}(3,3) = 6$. Recall that $\pi : E_H \to [K]$ is an arbitrary $1 : 1$ mapping from the set of pairs $(\ell_i, r_j) \in E_H$ to $[K]$. In this example, we choose $\pi(\ell_1, r_1) = 1, \pi(\ell_1, r_2) = 2, \pi(\ell_2, r_1) = 3, \pi(\ell_2, r_3) = 4, \pi(\ell_3, r_2) = 5$ and $\pi(\ell_3, r_3) = 6$. Furthermore, in this example we have $z = 010010$. Hence, since H^z is obtained from H by keeping only the edges that correspond to the 0 bits in z, we have that the only edges in H^z are $(\ell_1^z, r_1^z), (\ell_2^z, r_1^z), (\ell_2^z, r_3^z)$ and (ℓ_3^z, r_3^z).

$E_{H^z} = \{(\ell_i^z, r_j^z) \mid (\ell_i, r_j) \in E_H \wedge z_{\pi(\ell_i, r_j)} = 0\}$. See Figure 3.2 for an illustration of obtaining H^z from H and an input string $z \in \{0,1\}^K$.

Description of $\widetilde{H}^z$: $\widetilde{H}^z$ is obtained from H^z by replacing each edge $(\ell_i^z, r_j^z) \in E_{H^z}$ with a path of $b+1$ nodes and b edges, starting at ℓ_i^z and ending at r_j^z, where b is some positive integer to be chosen later. We denote this path by $P^z_{(\ell_i^z, r_j^z)}$. We slightly abuse notation and denote the set of nodes on this path also by $P^z_{(\ell_i^z, r_j^z)}$. We sometimes treat $P^z_{(\ell_i^z, r_j^z)}$ as a set of nodes, and sometimes as a path, but this will be clear from the context. Hence, the set of nodes of $\widetilde{H}^z$ is

$$V_{\widetilde{H}^z} = L^z \cup R^z \cup \bigcup_{(\ell_i^z, r_j^z) \in E_{H^z}} P^z_{(\ell_i^z, r_j^z)}$$

and the edges of $\widetilde{H}^z$ are only the ones on the paths in $\{P^z_{(\ell_i^z, r_j^z)} \mid (\ell_i^z, r_j^z) \in E_{H^z}\}$. Observe that $\widetilde{H}^z$ is not necessarily bipartite. See also Figure 3.3 for an illustration of how to obtain $\widetilde{H}^z$ from H^z.

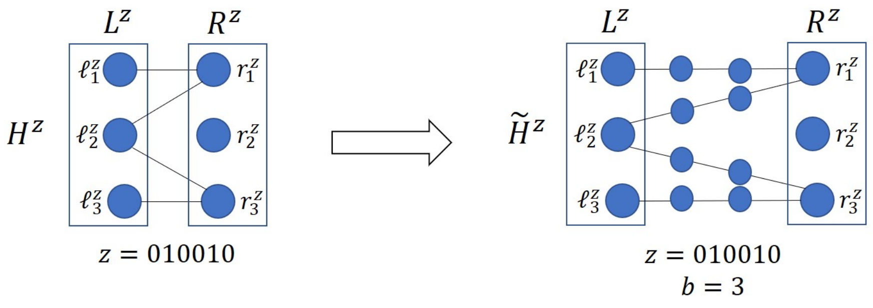

Figure 3.3: An illustration of how to obtain $\widetilde{H}^z$ from H^z for $b = 3$. The other parameters in this example are exactly as the ones chosen for Figure 3.2. That is, $t = p = 3$ and $\pi(\ell_1, r_1) = 1, \pi(\ell_1, r_2) = 2, \pi(\ell_2, r_1) = 3, \pi(\ell_2, r_3) = 4, \pi(\ell_3, r_2) = 5$ and $\pi(\ell_3, r_3) = 6$. $\widetilde{H}^z$ is obtained from H^z by simply stretching each edge in H^z to a path of length b.

Obtaining $G_{(x,y)}$ from $\widetilde{H}^x$ and $\widetilde{H}^y$: Given two input strings $x, y \in \{0,1\}^K$, $G_{(x,y)}$ is composed of $\widetilde{H}^x$ and $\widetilde{H}^y$ where we add a perfect matching between L^x and L^y, $\{(\ell_i^x, \ell_i^y) \mid i \in [p]\}$, and a perfect matching between R^x and R^y, $\{(r_i^x, r_i^y) \mid i \in [p]\}$. This concludes our construction. See also figures 3.4 and 3.5 for illustrations of $G_{(x,y)}$. In these figures, we also illustrate $\widetilde{H}^z$ for $z = x \wedge y$, where the string $x \wedge y \in \{0,1\}^K$ is defined by $(x \wedge y)_h = x_h \cdot y_h$ for any $h \in [K]$. That is, $(x \wedge y)_h = 1$ if and only if $x_h = y_h = 1$. The reason that we illustrate $\widetilde{H}^z$ in the same figures is that our proof heavily relies on comparing distances in $G_{(x,y)}$ to distances in $\widetilde{H}^z$ for $z = x \wedge y$. Figure 3.4 is an illustration of the two graphs when the strings are not disjoint, while Figure 3.5 is an illustration of the two graphs when the strings are disjoint. Before we prove that $\{\, G_{(x,y)} \mid x, y \in \{0,1\}^K \,\}$ is a family of lower bound graphs, we show the following two useful properties of the balanced bipartite graph $H = (L, R, E_H)$ that was described above.

Property 1. *If t is odd, then the distance between any two nodes $u, v \in L$ in H is at most $t - 1$. Similarly, the distance between any two nodes $u, v \in R$ in H is at most $t - 1$.*

Proof. The distance between every two nodes in H is at most its diameter t, but the distance between every two nodes in the same side of the bi-partition is even, so it is at most $t - 1$. □

Property 2. *If t is even, then the distance between any pair of nodes $u \in L$ and $v \in R$ in H is at most $t-1$.*

Proof. The distance between every two nodes in H is at most its diameter t, and the distance between two nodes in different sides of the bi-partition is odd, so it is at most $t-1$. □

3.4.2 $G_{(x,y)}$ is a family of lower bound graphs

Our goal in this section is to prove that $\{G_{(x,y)} \mid x, y \in \{0,1\}^K\}$ is a family of lower bound graphs with respect to the Set-Disjointness function and the graph predicate that distinguishes graphs of diameter $t(b+1)+1$ from graphs of diameter $(2t-1)b$. Let $z = x \wedge y$. Our proof relies on comparing distances between nodes in $G_{(x,y)}$ to distances between nodes in $\widetilde{H}^z$. While the proof contains many technical details that require some care, it follows from the following simple intuition.

Intuition and overview of the proof: First, it is not very hard to see that the diameter of $G_{(x,y)}$ is roughly equal to the diameter of $\widetilde{H}^z$ (up to an additive $t+1$). Hence, it suffices to argue that if the strings x and y are disjoint, then the diameter of $\widetilde{H}^z$ is at most tb, and otherwise the diameter of $\widetilde{H}^z$ is at least $(2t-1)b$. The main idea is to note that H^z is isomorphic to H if and only if the strings x and y are disjoint. Hence, if the strings are disjoint, the diameter of H^z is equal to the diameter of H which is t, and since $\widetilde{H}^z$ is obtained from H^z by stretching each edge to a path of length b, the diameter of $\widetilde{H}^z$ is tb.

On the other hand, if H^z is not isomorphic to H, then there is an edge in H for which the corresponding edge in H^z doesn't exist. Since the girth of H is $2t$, it implies that there are two nodes in H^z at distance at least $(2t-1)$ from each other. Hence, the distance between the corresponding two nodes in $\widetilde{H}^z$ is at least $(2t-1)b$. Next, we formalize these ideas and give a more detailed proof. The non-disjointness case is proved in Lemma 3.4.5, which uses claims 3.4.2, 3.4.3 and 3.4.4. The disjointness case is proved in Lemma 3.4.9, which uses claims 3.4.6, 3.4.7, and 3.4.8. Recall that given a graph G and two nodes u, v in it, we denote by $d_G(u,v)$ the distance between u and v in G.

Non-disjointness case:

Claim 3.4.2. If x and y are not disjoint, then the diameter of H^z is at least $2t-1$ and the diameter of $\widetilde{H}^z$ is at least $(2t-1)b$. In particular, there are $\ell_i^z \in L^z$ and $r_j^z \in R^z$ such that $d_{\widetilde{H}^z}(\ell_i^z, r_j^z) \geqslant (2t-1)b$.

Proof. Observe that if the strings x and y are not disjoint, then there is an $h \in [K]$ for which it holds that $x_h = y_h = 1$. Hence, $z_h = 1$. Since H^z is obtained from H by keeping only the edges that correspond to the 0 bits in z, it follows that there is an edge $(\ell_i, r_j) \in H$ such that there is no edge between the corresponding pair (ℓ_i^z, r_j^z) in H^z. Hence, since H has girth $2t$, if follows that the distance between ℓ_i^z and r_j^z in H^z is at least $2t-1$. Since $\widetilde{H}^z$ is obtained from H^z by replacing each edge with a path of length b, it follows that the diameter of $\widetilde{H}^z$ is at least $(2t-1)b$. $\square$

Claim 3.4.3. For any $(\ell_i, r_j) \in E_H$, if one of the paths $P^x_{(\ell_i^x, r_j^x)}$ and $P^y_{(\ell_i^y, r_j^y)}$ exists in $G_{(x,y)}$, then the path $P^z_{(\ell_i^z, r_j^z)}$ exists in $\widetilde{H}^z$. Similarly, if $P^z_{(\ell_i^z, r_j^z)}$ exists in $\widetilde{H}^z$, then either $P^x_{(\ell_i^x, r_j^x)}$ exists in $G_{(x,y)}$ or $P^y_{(\ell_i^y, r_j^y)}$ exists in $G_{(x,y)}$.

Proof. Let $h = \pi(\ell_i, r_j)$. Observe that if one of the paths $P^x_{(\ell_i^x, r_j^x)}$ and $P^y_{(\ell_i^y, r_j^y)}$ exists in $G_{(x,y)}$, then it must be the case that either $x_h = 0$ or $y_h = 0$. Hence, $z_h = 0$. Therefore there is an edge between ℓ_i^z and r_j^z in H^z, which is stretched to a path $P^z_{(\ell_i^z, r_j^z)}$ in $\widetilde{H}^z$. The other direction of the claim is proved similarly. $\square$

Claim 3.4.4. For any $\ell_i^x \in L^x$ and $r_j^x \in R^x$, it holds that $d_{G_{(x,y)}}(\ell_i^x, r_j^x) \geqslant d_{\widetilde{H}^z}(\ell_i^z, r_j^z)$.

Proof. Consider a shortest path between ℓ_i^x and r_j^x in $G_{(x,y)}$. Observe that this path is composed of edges crossing from $\widetilde{H}^x$ to $\widetilde{H}^y$ or vice versa (i.e., edges in $(L^x \times L^y) \cup (R^x \times R^y)$), and of paths of length b crossing from $L^x \cup L^y$ to $R^x \cup R^y$ or vice versa. Let q be the number of paths of length b crossing from $L^x \cup L^y$ to $R^x \cup R^y$ (or vice versa) that are used by the shortest path. And denote these paths by $P^1, P^2, \cdots, P^q$. Clearly, $d_{G_{(x,y)}}(\ell_i^x, r_j^x) \geqslant qb$. Hence, it suffices to show that $qb \geqslant d_{\widetilde{H}^z}(\ell_i^z, r_j^z)$.

For this, observe that for any $h \in [q]$, there are $ih, jh \in [p]$ and $w \in \{x, y\}$, for which $P^h = P^w_{(\ell_{ih}^w, r_{jh}^w)}$ (That is, P^h is connecting either a pair $(\ell_{ih}^x, r_{jh}^x) \in L^x \times R^x$ or a pair $(\ell_{ih}^w, r_{jh}^w) \in L^y \times R^y$). Hence, by Claim 3.4.3, this implies that for any $h \in [q]$, the path $P^z_{(\ell^z_{ih}, r^z_{jh})}$ exists in $\widetilde{H}^z$. Therefore, by starting at ℓ^z_i and following these q paths of length b in $\widetilde{H}^z$ we reach r^z_j. Hence, $qb \geqslant d_{\widetilde{H}^z}(\ell^z_i, r^z_j)$. □

Lemma 3.4.5. *If x and y are not disjoint, then the diameter of $G_{(x,y)}$ is at least $(2t-1)b$.*

Proof. By Claim 3.4.2, if the strings are not disjoint, then there are $\ell^z_i \in L^z$ and $r^z_j \in R^z$ such that $d_{\widetilde{H}^z}(\ell^z_i, r^z_j) \geqslant (2t-1)b$. Furthermore, by Claim 3.4.4, it holds that $d_{G_{(x,y)}}(\ell^x_i, r^x_j) \geqslant d_{\widetilde{H}^z}(\ell^z_i, r^z_j)$. Hence, there are two nodes in $G_{(x,y)}$ at distance at least $(2t-1)b$ from each other. □

Disjointness case:

Claim 3.4.6. If x and y are disjoint, then H^z is isomorphic to H. In particular, this implies:

1. H^z has diameter t.
2. for odd values of t, and for any two nodes $u, v \in V_{\widetilde{H}^z}$, if $u, v \in L^z$ or $u, v \in R^z$, then $d_{\widetilde{H}^z}(u, v) = (t-1)b$.
3. for even values of t, and for any two nodes $u, v \in V_{\widetilde{H}^z}$ such that $u \in L^z$ and $v \in R^z$, it holds that $d_{\widetilde{H}^z}(u, v) = (t-1)b$.

Proof. Observe that if the strings x and y are disjoint, then for any $h \in [K]$, either $x_h = 0$ or $y_h = 0$, so $z = x \wedge y$ is the all-zero string. Therefore, since H^z is obtained from H by keeping the edges that correspond to the 0 bits in z, H^z is isomorphic to H. Since H has diameter t, H^z also has diameter t.

Moreover, since H^z is isomorphic to H, by Property 1, it holds that for odd values of t, and for u, v that are on the same side (i.e., $u, v \in L^z$ or $u, v \in R^z$), it holds that $d_{H^z}(u, v) = t - 1$, and therefore $d_{\widetilde{H}^z}(u, v) = (t-1)b$.

Similarly, by Property 2, it holds that for even values of t, and for u, v that are not on the same side (i.e., $u \in L^z$ and $v \in R^z$), it holds that $d_{H^z}(u, v) = t - 1$, and therefore $d_{\widetilde{H}^z}(u, v) = (t-1)b$. □

For two nodes $u, v \in L^x \cup L^y \cup R^x \cup R^y$ in $G_{(x,y)}$, we say that u and v are on the same side if $u, v \in L^x \cup L^y$ or $u, v \in R^x \cup R^y$. Similarly, we say that u and v are on different sides if $u \in L^x \cup L^y$ and $v \in R^x \cup R^y$.

Claim 3.4.7. For odd values of t, if x and y are disjoint then for any two nodes $u, v \in L^x \cup L^y \cup R^x \cup R^y$ that are on the same side (i.e., either $u, v \in L^x \cup L^y$ or $u, v \in R^x \cup R^y$) it holds that $d_{G_{(x,y)}}(u, v) \leqslant (t-1)b + t$.

Proof. We prove the claim for $u, v \in L^x$. The other cases have a similar proof. If $u, v \in L^x$, there is some $i \in [p]$ for which $u = \ell_i^x$, and some $j \in [p]$ for which $v = \ell_j^x$. By the second item of Claim 3.4.6, it holds that $d_{\widetilde{H}^z}(\ell_i^z, \ell_j^z) = (t-1)b$. Hence, there is a shortest path between ℓ_i^z and r_j^z in $\widetilde{H}^z$ composed of exactly $(t-1)$ paths of length b in $\{P^z_{(\ell^z_{i'}, r^z_{j'})} \mid (\ell^z_{i'}, r^z_{j'}) \in E_{H^z}\}$. Denote these paths by $P^1, P^2, \cdots, P^{t-1}$. Observe that for any $h \in [t-1]$, there are $ih, jh \in [p]$ such that $P^h = P^z_{(\ell^z_{ih}, r^z_{jh})}$, where $i1 = i$. Therefore, by Claim 3.4.3, it holds that either $P^x_{(\ell^x_{ih}, r^x_{jh})}$ exists in $G_{(x,y)}$ or $P^y_{(\ell^y_{ih}, r^y_{jh})}$ exists in $G_{(x,y)}$ for any $h \in [t-1]$.

Hence, one can use these $t-1$ paths in $G_{(x,y)}$ to reach from ℓ_i^x to ℓ_j^x as follows. We start at ℓ_i^x and if $P^x_{(\ell^x_i, r^x_{j1})}$ exists in $G_{(x,y)}$, then we use it to move to r^x_{j1}. Otherwise, it must be the case that $P^y_{(\ell^y_i, r^y_{j1})}$ exists in $G_{(x,y)}$ (Recall that $i1 = i$). Hence we can use the edge (ℓ_i^x, ℓ_i^y) to move to ℓ_i^y and then we can use the path $P^y_{(\ell^y_i, r^y_{j1})}$ to move to r^y_{j1}.

We can keep alternating in this way between $\widetilde{H}^x$ and $\widetilde{H}^y$ to reach ℓ_j^x, where the total number of edges in $(L^x \times L^y) \cup (R^x \times R^y)$ that we use is at most t. And the number of paths of length b that we use is exactly $t-1$. Hence, in total, the length of this path is at most $(t-1)b + t$. □

Claim 3.4.8. For even values of t, if x and y are disjoint then for any two nodes $u, v \in L^x \cup L^y \cup R^x \cup R^y$ that are on different sides (i.e., $u \in L^x \cup L^y$ and $v \in R^x \cup R^y$) it holds that $d_{G_{(x,y)}}(u, v) = (t-1)b + t$.

Proof. We prove the claim for $u \in L^x$ and $v \in R^x$. The other cases have a similar proof. Hence, there are $i, j \in [p]$ such that $u = \ell_i^x$ and $v = r_j^x$. The proof is almost identical to the proof of Claim 3.4.7. The only difference is that in the proof of Claim 3.4.7 we used the second item of Claim 3.4.6 to argue that $d_{\widetilde{H}^z}(\ell_i^z, \ell_j^z) = (t-1)b$. Here, we are dealing with even values of t. Therefore, instead of the second item of Claim 3.4.6, we use the third item of the same claim which deals with even value of t. By the third item of Claim 3.4.6, it holds that $d_{\widetilde{H}^z}(\ell_i^z, r_j^z) = (t-1)b$. The rest of the proof is identical to the proof of Claim 3.4.7. □

Lemma 3.4.9. *If x and y are disjoint, then the diameter of $G_{(x,y)}$ is at most $tb + t + 1$.*

Proof. Let u, v be two nodes in $G_{(x,y)}$. Let a_u^ℓ be the distance from u to the closest node in $L^x \cup L^y$, and let a_u^r be the distance from u to the closest node in $R^x \cup R^y$. a_v^ℓ and a_v^r are defined similarly for v. For example, if $u \in L^x \cup L^y$ then $a_u^\ell = 0$. The key point to note is that either $a_u^\ell + a_v^\ell \leqslant b + 1$, or $a_u^r + a_v^r \leqslant b + 1$. That is, either taking the two nodes to the left side of $G_{(x,y)}$ (i.e., to $L^x \cup L^y$) costs at most $b+1$, or taking the two nodes to the right side costs at most $b+1$. Similarly, it holds that either $a_u^\ell + a_v^r \leqslant b + 1$, or $a_u^r + a_v^\ell \leqslant b + 1$. The rest of the proof is by the following case analysis.

1. t is odd: By Claim 3.4.7, the distance between any two nodes in $L^x \cup L^y$ and the distance between any two nodes in $R^x \cup R^y$ is at most $(t-1)b + t$. Furthermore, we can either move u and v to $L^x \cup L^y$ by using at most $b+1$ steps (in total, for moving both u and v), or we can move u and v to $R^x \cup R^y$ by using at most $b+1$ steps (in total, for moving both u and v). After moving u and v to one of the sides, we can use Claim 3.4.7 and deduce that $d_{G_{(x,y)}}(u,v) \leqslant (t-1)b + t + b + 1 = tb + t + 1$.

2. t is even: By Claim 3.4.8, the distance between any $u' \in L^x \cup L^y$ and $v' \in R^x \cup R^y$ is at most $(t-1)b + t$. Furthermore, we can either move u to $L^x \cup L^y$ and v to $R^x \cup R^y$ by using at most $b+1$ steps (in total, for moving both u and v), or we can move u to $R^x \cup R^y$ and v to $L^x \cup L^y$ by using at most $b+1$ steps (in total, for moving both u and v). Hence, we can use Claim 3.4.8 and deduce that $d_{G_{(x,y)}}(u,v) \leqslant (t-1)b + t + b + 1 = tb + t + 1$. □

3.4.3 Proof of Theorem 3.4.1

Proof. First, we define the following partition $V = V_A \dot\cup V_B$ of the set of nodes of $G_{(x,y)}$. $V_A = V_{\widetilde{H}^x}$, and $V_B = V_{\widetilde{H}^y}$. Hence, the size of the cut $C = E(V_A, V_B)$ is $\Theta(p)$. This is because the only edges connecting between nodes in $\widetilde{H}^x$ and nodes in $\widetilde{H}^y$ in $G_{(x,y)}$ are the $2p$ edges of the matching between L^x and L^y, and the matching between R^x and R^y.

Since our goal is to show a lower bound as a function of the number of nodes n in the input graph $G_{(x,y)}$, we need to analyze the size of the cut and the size of the input strings with respect to n. By Theorem 3.2.7, we have that for $t \in \{3,4,6\}$, $K = \mathrm{Ex}(p,t) = \Omega(p^{1+\frac{1}{t-1}})$. Furthermore, by Remark 3.2.3 we can assume that a constant fraction of the bits in the strings x and y are 0. Hence, these 0 bits are translated to paths of length b in $G_{(x,y)}$. This implies that the number of nodes in $G_{(x,y)}$ is $n = \Theta(Kb) = \Omega(p^{1+\frac{1}{t-1}}b) = \Omega(p^{1+\frac{1}{t-1}})$ for constant values of b. Therefore, the size of the cut C is $\Theta(p) = O(n^{\frac{t-1}{t}})$.

Lemmas 3.4.5 and 3.4.9 imply that $\{G_{(x,y)} \mid x,y \in \{0,1\}^K\}$ is a family of lower bound graphs with respect to the Set-Disjointness function and the graph predicate that distinguishes between graphs of diameter $tb+t+1$ and graphs of diameter $(2t-1)b$. Hence, by Theorem 3.2.5 and the fact that the communication complexity of Set-Disjointness is $\Omega(K)$, we have that any algorithm for distinguishing between these two cases in the CONGEST model requires $\Omega\left(\frac{K}{|C|\log n}\right) = \Omega\left(\frac{n}{n^{(t-1)/t}\log n}\right) = \Omega\left(\frac{n^{1/t}}{\log n}\right)$ rounds.

To get this lower bound for $(\frac{t}{2t-1}+\epsilon)$-approximation for diameter, we need that $\left(\frac{t}{2t-1}+\epsilon\right)(2t-1)b > tb+t+1$. Hence, we can pick $b = \Theta\left(\frac{t+1}{\epsilon(2t-1)}\right) = \Theta\left(\frac{1}{\epsilon}\right)$. □

3.4.4 Handling connectivity

One may wonder about the connectivity of the graph $G_{(x,y)}$. As the construction was described so far, there could be some values of x and y such that $G_{(x,y)}$ is not connected. In this section, we show how to slightly modify the construction of $G_{(x,y)}$ so that it is always connected, and in fact of constant diameter, without changing the analysis. Observe that it suffices to make $\widetilde{H}^x$ always connected. This is because any node in $\widetilde{H}^y$ has some path connecting it to a node in $\widetilde{H}^x$. Since $\widetilde{H}^x$ is obtained from H^x by stretching each edge in H^x to a path of length b, it suffices to make H^x always connected, regardless of the input string x.

Recall that given a string $x \in \{0,1\}^K$, we defined H^x to be the graph obtained from H where we keep only the edges that correspond to the 0 bits in x. Recall that H is a balanced bipartite graph of size $2p$, diameter t and girth $2t$. Of course, H is always connected. But since some of the edges of H may not exist in H^x, H^x may not be connected. To ensure that H^x is connected, let S be a shortest paths tree starting from an arbitrary node in H. Of course, the number of edges in S is $O(p)$, which is small with respect to the size of the input string x which is $K = \mathrm{Ex}(p, t)$.

We modify the definition of H^x such that the edges that correspond to the spanning tree S always exist in H^x. In particular, their existence in H^x doesn't depend on x. For this, we need to modify the size of the string x to $K - |S| = \Theta(K)$, so that only the edges that are not in S depend on x. The proof that $\{G_{(x,y)} \mid (x, y) \in \{0,1\}^{K-|S|} \times \{0,1\}^K\}$ is a family of lower bound graphs remains exactly the same as in Section 3.4.2. Furthermore, since the size of x didn't change asymptotically, the deduced lower bound from Section 3.4.3 doesn't change asymptotically. Observe that under the new definition of H^x, the diameter of H^x is at most $2t$. Hence, the diameter of $\widetilde{H}^x$ is at most $2tb$. It is not very hard to verify that the diameter of $G_{(x,y)}$ in this case is at most $2tb + 2b + 2$, which is constant for constant values of t and b.

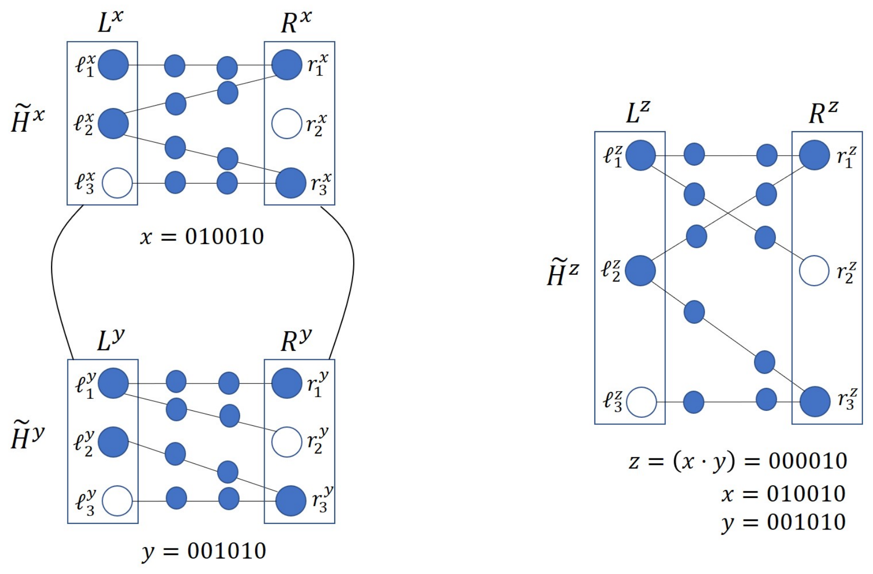

Figure 3.4: An Illustration of $G_{(x,y)}$ (on the left) and $\widetilde{H}^z$ for $z = x \wedge y$ (on the right). $G_{(x,y)}$ is composed of $\widetilde{H}^x$ and $\widetilde{H}^y$, where we add a perfect matching between L^x and L^y, $\{(\ell_i^x, \ell_i^y) \mid i \in [p]\}$, and a perfect matching between R^x and R^y, $\{(r_i^x, r_i^y) \mid i \in [p]\}$. The edge between L^x and L^y in this figure represents the matching between them. Similarly, the edge between R^x and R^y in this figure represents the matching between them. The parameters in this example are exactly as the ones chosen for Figures 3.2 and 3.3. That is, $t = p = 3$, $b = 3$, $\pi(\ell_1, r_1) = 1, \pi(\ell_1, r_2) = 2, \pi(\ell_2, r_1) = 3, \pi(\ell_2, r_3) = 4, \pi(\ell_3, r_2) = 5$ and $\pi(\ell_3, r_3) = 6$. Furthermore, we have $x = 010010, y = 001010$ and $z = x \wedge y = 000010$. Hence, the only paths of length b that we have in $\widetilde{H}^x$ are $P^x_{(\ell_1^x, r_1^x)}, P^x_{(\ell_2^x, r_1^x)}, P^x_{(\ell_2^x, r_3^x)}$ and $P^x_{(\ell_3^x, r_3^x)}$. And the only paths of length b that we have in $\widetilde{H}^y$ are $P^y_{(\ell_1^y, r_1^y)}, P^y_{(\ell_1^y, r_2^y)}, P^y_{(\ell_2^y, r_3^y)}$ and $P^y_{(\ell_3^y, r_3^y)}$. Observe that the path $P^x_{(\ell_3^x, r_2^x)}$ doesn't exist in $\widetilde{H}^x$, and that the path $P^y_{(\ell_3^y, r_2^y)}$ doesn't exist in $\widetilde{H}^y$. This is because $x_{\pi(\ell_3, r_2)} = x_5 = 1$ and $y_{\pi(\ell_3, r_2)} = y_5 = 1$. This implies that $z_{\pi(\ell_3, r_2)} = z_5 = 1$ as well, and therefore the path $P^z_{(\ell_3^z, r_2^z)}$ doesn't exist in $\widetilde{H}^z$. It is easy to see that the distance between ℓ_3^z and r_2^z in $\widetilde{H}^z$ is $5b = (2t-1)b$. One can verify that the distance between ℓ_3^x and r_2^x in $G_{(x,y)}$ is at least $5b = (2t-1)b$ as well. This illustrates that when the strings x and y are disjoint, the diameter of $G_{(x,y)}$ is at least $(2t-1)b$.

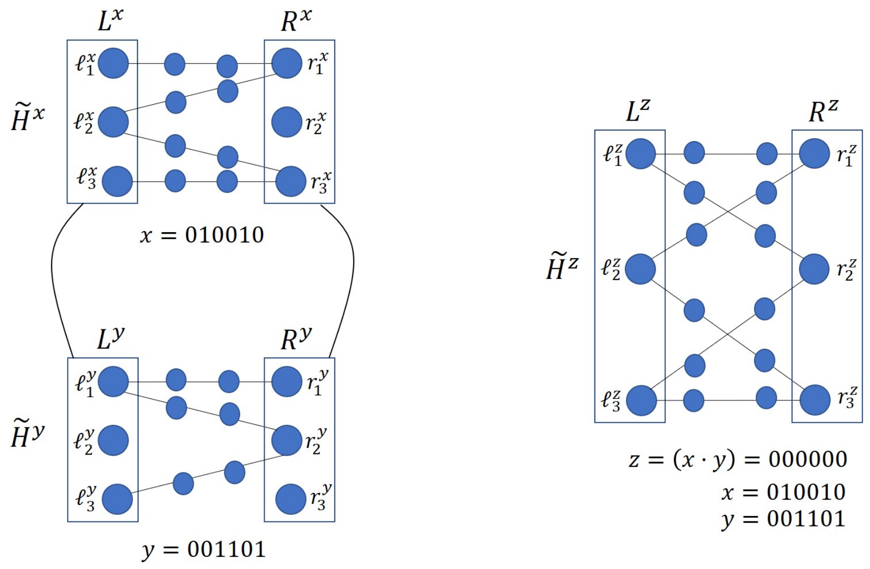

Figure 3.5: An illustration of $G_{(x,y)}$ and $\widetilde{H}^z$ for $z = x \wedge y$. The parameters in this example are exactly as the ones chosen for Figures 3.2, 3.3, and 3.4. The only difference in this example compared to the one in Figure 3.4 is that the strings x and y are disjoint. Hence, $z = x \wedge y$ is an all zeros string. It is easy to see that in this case the diameter of $\widetilde{H}^z$ is tb. The key point of the proof is that the diameter of $G_{(x,y)}$ is not very much larger than the diameter of $\widetilde{H}^z$ (in fact, it is larger by at most $t + 1$, which is negligible compared to tb for values of $b \gg t$). To illustrate this in this example, we show a path of length $tb + t = 3b + 3$ from ℓ_1^x and r_3^y. We start by moving from ℓ_1^x to ℓ_1^y using the edge (ℓ_1^x, ℓ_1^y) that is part of the matching between L^x and L^y. Then, we use the path of length b from ℓ_1^y to r_2^y, and the path of length b from r_2^y to ℓ_3^y. After that, we use the edge (ℓ_3^y, ℓ_3^x) to move to ℓ_3^x, and the path of length b from ℓ_3^x to r_3^x. Finally, we use the edge (r_3^x, r_3^y) to reach r_3^y. This example illustrates that the diameter of $G_{(x,y)}$ is relatively small if the strings are disjoint.

Part II

Distributed Symmetry Breaking

Chapter 4

Distributed Approximate Maximum Independent Set

This chapter is based on joint work with Ken-ichi Kawarabayashi, Aaron Schild, and Gregory Schwartzman [197], in which we show new upper and lower bounds for approximate Maximum Independent Set in the distributed model.

4.1 Introduction

We study the problem of finding an approximately maximum independent set (MaxIS) in the LOCAL and CONGEST models of Distributed Computing [43, 80, 103, 121, 145, 163, 173, 208]. In unweighted graphs, one can find a Δ-approximation for MaxIS by finding a *maximal* independent set (MIS), where Δ is the maximum degree of a node in the graph. In recent years, our understanding of the complexity of MIS has been substantially improving [63, 157, 158, 247], leading to a remarkable breakthrough by Rozhon and Ghaffari [247], where they show a deterministic $\text{poly}(\log n)$-round algorithm for finding an MIS, even in the CONGEST model [104]. This result also implies a randomized algorithm that succeeds with high probability in $O(\log \Delta + \text{poly}(\log \log n))$ rounds in the CONGEST model[1] [104, 158, 247].

In a weighted graph, an MIS doesn't necessarily constitute a Δ-approximation for MaxIS. For the weighted case, Bar-Yehuda et al. [56] showed a Δ-approximation

[1]We say that an algorithm succeeds with high probability if it succeeds with probability $1 - 1/n^c$ for an arbitrary constant $c > 1$.

algorithm in the CONGEST model that takes $O(MIS(n, \Delta) \cdot \log W)$ rounds, where $MIS(n, \Delta)$ is the running time for finding an MIS in graphs with n nodes and maximum degree Δ, and W is the maximum weight of a node in the graph (which can be as high as $\text{poly}(n)$). Whether their algorithm is deterministic or randomized depends on the MIS algorithm that is used as a black box.

In this work, we present faster algorithms compared to [56], by paying only a $(1 + \epsilon)$ multiplicative overhead in the approximation factor. Our main result (Theorem 4.1.2) is a randomized algorithm that achieves an exponential speed-up compared to [56].

Theorem 4.1.1. *There is an $O(MIS(n, \Delta)/\epsilon)$-round algorithm in the CONGEST model that finds a $(1 + \epsilon)\Delta$-approximation for maximum-weight independent set. Whether the algorithm is deterministic or randomized, depends on the MIS algorithm that is run as a black-box.*

Theorem 4.1.2. *There is a randomized $(\text{poly}(\log \log n)/\epsilon)$-round algorithm in the CONGEST model that finds, with high probability, a $(1 + \epsilon)\Delta$-approximation for maximum-weight independent set.*

Due to a lower bound of $\Omega(\sqrt{\log n / \log \log n})$ that was given by Kuhn, Moscibroda and Wattenhofer [204], against any (possibly randomized) algorithm that finds an MIS, even in the LOCAL model, Theorem 4.1.2 implies that finding a $(1 + \epsilon)\Delta$-approximation for MaxIS is exponentially easier than MIS.

Results for unweighted graphs: It is easy to see that a single round of Luby's algorithm [216] yields a solution with an *expected* weight at least $w(V)/(\Delta + 1)$, where $w(V)$ is the total weight of nodes in the graph.[2] In this algorithm, each node v picks a number r_v uniformly at random in $[0, 1]$. If $r_v > r_u$ for any neighbor u of v, then v joins the independent set. Since every node joins the independent set with probability at least $1/(\Delta + 1)$, the expected weight of the independent set is at least $w(V)/(\Delta + 1)$. However, algorithms that work well in expectation don't necessarily work well with good probability. In fact, for a single-round of Luby's algorithm, it is not very hard to construct examples in which the *variance* of the solution is very high, in which case the algorithm doesn't return the expected

[2]This single-round algorithm is also known as the ranking algorithm. To the best of our knowledge, the classical ranking algorithm has first appeared in the book of Alon and Spencer [29] and is due to Boppana (see also the references for this algorithm in [85]).

value with high probability. In this work we prove the following stronger theorem for any algorithm.

Theorem 4.1.3. *In the LOCAL model, assuming that the nodes don't know the exact value of n, but only a polynomial upper bound on it, any algorithm that finds an independent set of size $\Omega(n/\Delta)$ in unweighted graphs of n nodes and maximum degree $\Delta \in \{\Theta(n/\log n), \Theta(n/\log^* n)\}$, with success probability $p \geqslant 1 - 1/(10\log n)$, must spend $\Omega(\log^* n)$ rounds. The lower bound holds even if the nodes know the exact value of Δ.*

In our lower bound, we exploit the fact that a general algorithm must succeed on any input graph (with the desired success probability). In particular, it must succeed on graphs of maximum degree $\Theta(n/\log^* n)$, and it must succeed on graphs of maximum degree $\Theta(n/\log n)$ (since a general algorithm must succeeds on any graph, it must also succeed on graphs with maximum degree in $\{\Theta(n/\log n), \Theta(n/\log^* n)\}$). One may wonder whether the lower bound still applies when the input graph is guaranteed to have a smaller maximum degree (as a function of n). We rule out this possibility, with the following theorem. The proof of Theorem 4.1.4 relies on a novel idea for analyzing the classical ranking algorithm using martingales and the local-ratio technique, on which we elaborate in the technical overview.

Theorem 4.1.4. *For unweighted graphs of maximum degree $\Delta \leqslant n/\log n$, there is an $O(1/\epsilon)$-round algorithm in the CONGEST model that finds, with high probability, an independent set of size at least $\frac{n}{(1+\epsilon)(\Delta+1)}$.*

Road-map: In Section 4.2 we provide a technical overview. Section 4.3 contains some basic definitions and useful inequalities. In Section 4.4, we prove our first two results (Theorems 4.1.1 and 4.1.2). Our results for low-degree graphs are presented in Sections 4.5. Our lower bound result is presented in Section 4.6.

4.2 Technical Overview

Results for weighted graphs: Our first two results (Theorems 4.1.1 and 4.1.2) share a similar proof structure. First, we show that there are fast algorithms for $O(\Delta)$-approximation. Then we use the *local-ratio* technique [55] to prove a general boosting theorem that takes a T-round algorithm for $O(\Delta)$-approximation, and use it as a black-box to output a $(1+\epsilon)\Delta$-approximation in $O(T/\epsilon)$ rounds. An overview of the local-ratio technique and the boosting theorem is provided in Section 4.2.2. The key ingredient to show a fast $O(\Delta)$-approximation algorithm is a new *weighted sparsification* technique, where we show that it suffices to find an independent set of a good approximation in a sparse subgraph. An overview of the weighted sparsification technique is provided in Section 4.2.1.

Results for unweighted graphs: Our upper bound for unweighted graphs of maximum degree $\Delta \leqslant n/\log n$ (Theorem 4.1.4) has a similar two-step structure as the first two results. We first show an $O(\Delta)$-approximation algorithm, and then we use the local-ratio technique to boost the approximation factor. For the $O(\Delta)$-approximation part, we show that running the classical one-round of Luby's algorithm for c rounds already returns an $O(\Delta)$-approximation for unweighted graphs of maximum degree $\Delta \leqslant n/\log n$, with probability $\approx 1 - 1/n^c$. The main technical ingredient for showing this result is a new analysis of the classical ranking algorithm using *martingales*. An overview of this result is provided in Section 4.2.3. Finally, in Section 4.2.4, we provide an overview of the lower bound result (Theorem 4.1.3).

4.2.1 Weighted Sparsification for $O(\Delta)$-Approximation

Let us first consider the unweighted case for simplicity. Let $G = (V, E)$ be an unweighted graph. We can find an $O(\Delta)$-approximation for MaxIS in G as follows. First, we sample a sparse subgraph H of G with the following properties. (1) The maximum degree Δ_H of H is small ($O(\log n)$). (2) The ratio between the number of nodes (n_H) and the maximum degree of H is at least as in G, up to a constant multiplicative factor. That is, $n_H/\Delta_H = \Omega(n/\Delta)$. Since any MIS in H has size at least $n_H/\Delta_H = \Omega(n/\Delta)$, it suffices to find an MIS in H, which take $MIS(n_H, \Delta_H) \leqslant MIS(n, \log n)$ rounds (recall that $MIS(n, \Delta)$ is the running time of finding an MIS in graphs of n nodes and maximum degree Δ). By the breakthrough of Rozhon and Ghaffari [247], $MIS(n, \log n) = O(\log\log n) + \text{poly}(\log\log n) = \text{poly}\log\log n$ rounds. Furthermore, sampling a subgraph with the aforementioned properties is almost trivial. Each node joins H with probability $\min\{\log n/\Delta, 1\}$, independently. It is not very hard to show, via standard Chernoff (Theorem 4.3.1) and Union Bound arguments, that H has the desired properties. While this approach is straightforward for the unweighted case, it runs into challenges when trying to apply it for the weighted case, as we explain next.

The challenge in weighted graphs: Perhaps the first thing that comes into mind when trying to extend the sampling technique to weighted graphs is to try to sample a sparse subgraph H with the following properties. (1) The maximum degree $\Delta_H = O(\log n)$. (2) The ratio between the *total weight* in H and the max degree of H is the same as in G, up to a constant multiplicative factor. That is $w(V_H)/\Delta_H = \Omega(w(V)/\Delta)$, where $w(V_H)$ is the total weight of nodes in H and $w(V)$ is the total weight of nodes in G. However, this approach runs into two challenges. The first challenge is that in the weighted case, an MIS doesn't necessarily constitute a Δ-approximation for MaxIS. Therefore, even if we are able to sample a subgraph H with the desired properties, running an MIS algorithm on H might result in an independent set of a very small weight. To overcome this challenge, we show a very simple $MIS(n, \Delta)$-round algorithm that finds an $O(\Delta)$-approximation. This algorithm runs an MIS algorithm on the subgraph induced by nodes that are relatively heavy, compared to their neighbors. Specifically, a node is considered relatively heavy compared to its neighbors, if it is of weight at least $\Omega(1/\Delta)$-fraction of the sum of weights of its neighbors. It is not very hard to show that this algorithm returns an independent set of total weight $\Omega(w(V)/\Delta)$, where $w(V)$ is the total weight of nodes in the graph. The proof of this argument is provided in Section 5.4.

Furthermore, another challenge is that the same sampling procedure doesn't work for the weighted case. In particular, if we sample each node with probability $p = \min\{(\log n)/\Delta, 1\}$, then *light*-weight nodes will have the same probability of joining H as *heavy*-weight nodes. Intuitively, we need to take the weights into account. For this, we boost the sampling probability of a node v by an additive factor of $w(v) \log n / w(V)$, where $w(v)$ is the weight of v and $w(V)$ is the total weight of nodes in the graph. In order to show that the sampled subgraph has the desired properties, it doesn't suffice to use standard Chernoff and Union-Bound arguments. Instead, we present a more involved analysis that uses Bernstein's inequality (Theorem 4.3.2). Observe that the nodes don't know the value $w(V)$. Therefore, we define a notion of *weighted degree* of a node, which is the sum of weights of its neighbors. We show that it suffices for a node v to use the maximum weighted degree in its neighborhood, instead of $w(V)$. The full argument is provided in Section 4.4.2.

4.2.2 Boosting the Approximation Factor using Local-Ratio

A useful technique for approximation algorithms is the local-ratio technique [55]. In recent years, the local-ratio technique has been found to be very useful for the distributed setting [56, 58], and the Δ-approximation algorithm of [56] also uses this technique. In this work we use local-ratio to boost the approximation guarantee for MaxIS. We start with stating the local-ratio theorem for maximization problems. Here, we state it specifically for MaxIS. Given a weighted graph $G_w = (V, E, w)$, where w is a node-weight function $w : V \to \mathbb{R}$, we say that an independent set $I \subseteq V$ is r-approximate with respect to w if it is r-approximate for the optimal solution in G_w.

Theorem 4.2.1. ***[Theorem 9 in [55]]***
Let $G_w = (V, E, w)$ be a weighted graph. Let w_1 and w_2 be two node-weight functions such that $w = w_1 + w_2$. If an independent set I is r-approximate with respect to w_1 and with respect to w_2 then it is r-approximate with respect to w as well.

Theorem 4.2.1 already gives a simple linear-time sequential algorithm for Δ-approximation for MaxIS, as follows. Pick an arbitrary node v of positive weight, push it onto a stack, and reduce the weight of any node in the inclusive neighborhood of v (v and its neighbors) by $w(v)$. Continue recursively on the obtained graph, until there are no nodes of positive weight. When there are no remaining nodes of positive weight, pop out the stack, and construct an independent set I greedily, as follows. For each node v that is popped out from the stack, add v to I, unless it already contains a neighbor of v.

The reason that this simple algorithm gives a Δ-approximation is as follows. Consider the first iteration, when the algorithm picks an arbitrary node v, pushes it onto a stack, and reduces the weight of any node in the inclusive neighborhood of v by $w(v)$. This first iteration implicitly defines two weight functions: the *reduced* weight function w_1, and the *residual* weight function w_2, where $w = w_1 + w_2$. That is, the reduced weigh of a node u in the first step is $w_1(u) = w(v)$ if it belongs to the inclusive neighborhood of v, and $w_1(u) = 0$ otherwise. The residual weight of a node u is the remaining weight $w_2(v) = w(v) - w_1(v)$. To prove that the algorithm returns a Δ-approximation, we can assume by reverse induction that I is a Δ-approximation with respect to the residual weight function w_2. Furthermore, the independent set is constructed in a way such that it must contain at least one node in the inclusive neighborhood of v, where the weight of this node with respect to w_1 is $w(v)$. Since the degree of v is at most Δ, and the value of the optimal solution with respect to w_1 is at most $\Delta w(v)$, it follows that I is also Δ-approximation with respect to the reduced weight function w_1. Hence, by the local-ratio theorem, the independent set is also a Δ-approximation with respect to $w = w_1 + w_2$.

One can extend this idea, and rather than picking a single node in each step, the algorithm can pick an arbitrary independent set I', push all the nodes in I' onto a stack, and perform local weight reductions in the inclusive neighborhood of any node in I'. The algorithm continues recursively on the obtained graph after the weight reductions, until there are no remaining nodes of positive weight. Then, the algorithm constructs an independent set I by popping out the stack and adding nodes in the stack to I greedily. Using a similar local-ratio argument, one can show that this algorithm also returns a Δ-approximation for MaxIS. The idea of picking an independent set rather than a single node in each step was used by [56] to show a Δ-approximation algorithm in $O(MIS(n, \Delta) \log W)$ rounds.

In this work, we prove a simple yet powerful property about the local-ratio technique. Specifically, we show that the total weight of the independent set I that is constructed in the pop-out stage (with respect to the original input weight function w), is at least the total weight of the nodes in the stack (with respect to the residual weight function at the time they were pushed onto the stack). That is, let S be set of nodes that are pushed onto the stack. For $v \in S$, let w_{i_v} be the residual weight of v at the time it was pushed onto the stack. We prove (Proposition 4.4.12 in Section 4.4.3) that $w(I) \geqslant \sum_{v \in S} w_{i_v}(v)$. We refer to this property as the *stack property*.

The stack property allows us to show a general boosting theorem, as follows. We use the local-ratio algorithm described above, where in each step we pick an independent set I' that is $(c\Delta)$-approximation for MaxIS, for some constant $c > 1$. Hence, intuitively, after $\approx c/\epsilon$ steps, the total weight in the stack should be at least $\frac{OPT(G_w)}{(1+\epsilon)\Delta}$, where $OPT(G_w)$ is the value of an optimal solution in the input graph G_w. The full argument of the boosting theorem is provided in Section 4.4.3.

Low-arboricity graphs: Moreover, the stack property allows us to show an improved approximation algorithm for low-arboricity graphs, as follows. In each step, we run a $(1+\epsilon)\Delta$-approximation algorithm on the subgraph induced by the nodes of degree at most 4α, where α is the arboricity of the graph. We push the nodes in the resulting independent set I' onto the stack, and perform local weight reduction in the neighborhoods of the nodes in I'. Then, we delete all the nodes of degree at most 4α, and continue recursively on the resulting graph. Finally, we construct an independent I by popping out the stack greedily. By a standard Markov argument, after $\log n$ push steps, the graph becomes empty. Furthermore, since in each step the algorithm finds a $(1+\epsilon)4\alpha$ approximation in the subgraph induced by the nodes of degree at most 4α, and this independent set is pushed onto the stack, we are able to use the stack property to show that the constructed independent set I is roughly of the same approximation for G_w. We refer the reader to [197] for our results for low-arboricity graphs.

4.2.3 Analysis of the Ranking Algorithm using Martingales

In this section we provide an overview of our result for unweighted graphs of maximum degree $\Delta \leqslant n/\log n$ (Theorem 4.1.4). First, we find an $O(\Delta)$ approximation, and then we use the boosting theorem to get a $(1+\epsilon)\Delta$-approximation. To find an $O(\Delta)$-approximation, we use the classical ranking algorithm. Recall that in the ranking algorithm, each node v picks a number r_v uniformly at random in $[0,1]$. If $r_v > r_u$ for any neighbor u of v, then v joins the independent set. Let I be the independent set that is returned by the ranking algorithm. The crux of the analysis is in using concentration inequalities to get a high-probability lower bound on the number of nodes in I. However, it is unclear how to make this approach work, as the random variables $X_v = \mathbf{1}_{v\in I}$ are not independent. While these random variables are not independent, one can obtain a weaker result in this direction. Specifically, for graphs of maximum degree at most $n^{1/3}/\text{poly}(\log n)$, one can get a useful bound on the maximum *dependency* among these variables. In particular, one can show that each X_v is dependent on at most $(n^{1/3}/\text{poly}(\log n))^2 = n^{2/3}/\text{poly}(\log n)$ other X_us, which makes it possible to show concentration using the bounded dependence Chernoff bound given in [239]. However, it is unclear how to use this approach for higher degree graphs.

The main idea of our approach is to view the ranking algorithm from a sequential perspective. Instead of picking ranks for the nodes and including a node in I if its rank is higher than that of its neighbors, we draw nodes v from V uniformly at random one at a time and add v to I if it is not adjacent to any previously drawn node. We show that the resulting independent set is identical in distribution to the independent set produced by the ranking algorithm (Proposition 4.5.1 in Section 4.5). Note that this is not the same as a sequential greedy algorithm for maximal independent set, which would add v to I if it is not adjacent to any node in I (a weaker condition). The sequential perspective of the ranking algorithm allows us to think about the size of I incrementally. One could directly show concentration if the family of random variables $\{I_t\}_t$ was a martingale. However, this is not the case, as $|I_{t+1}| \geqslant |I_t|$ so it is not possible for expected increments to be 0. Instead, we create a martingale by shifting the increments so that they have mean 0. More formally, let I_t be the independent set I after the first t nodes have been drawn. Let v_t be the tth node drawn. The random variable

$$Y_t = |I_t| - |I_{t-1}| - \mathbf{Pr}[v_t \in I | I_{t-1}]$$

has mean 0 conditioned on I_{t-1}. Therefore, the Y_ts are increments for the martingale $X_t = \sum_{i=1}^{t} Y_t$. Using Azuma's Inequality, one can show that X_t concentrates around its mean, which is 0. To lower bound the size of the obtained independent set I, one therefore just needs to get a lower bound on the sum of the increment probabilities $\mathbf{Pr}[v_t \in I | I_{t-1}]$. This can be lower bounded by $1/2$ when $t = o(n/\Delta)$ because when a node is drawn, it eliminates at most Δ other nodes from inclusion into I. But when $t = \Theta(n/\Delta)$, the sum of these probabilities is already $1/2(\Theta(n/\Delta)) = \Theta(n/\Delta)$, so the independent set is already large enough, as desired. The reason that this technique works for $\Delta \leqslant n/\log n$ is that the success probability is roughly exponential in n/Δ. Hence, by having $\Delta \leqslant n/\log n$, we get a high probability success, as desired. The full argument is provided in Section 4.5.

4.2.4 An Overview of the Lower Bound

In this section we give an overview of our lower bound (Theorem 4.1.3). For the deterministic case, one can show a lower bound for finding an independent set of size $\Omega(n/\Delta)$ in a cycle, by a reduction from the classical lower bound of Linial for finding an MIS in a cycle [213]. However, for the randomized case, this approach becomes more challenging. In fact, the cycle graph cannot be a hard instance for finding an independent set of size $\Omega(n/\Delta)$, as there is a constant-round algorithm for low degree graphs (as we show in Theorem 4.1.4). In order to show hardness for the randomized case, we use a *cycle of cliques* graph. We are able to reduce the problem of finding an independent set of size $\Omega(n/\Delta)$ in a cycle of cliques, to the problem of finding an MIS in a cycle. And we use Naor's lower bound [225] for finding an MIS in a cycle, which holds even against randomized algorithms. We start by stating Naor's lower bound.

Theorem 4.2.2. *(**Lower bound for the cycle** [225]). Any randomized algorithm in the LOCAL model for finding a maximal independent set that takes fewer than* $\frac{1}{2}(\log^* n) - 4$ *rounds, succeeds with probability at most* $1/2$*, even for a cycle of length* n*.*

Perhaps a good way to understand our reduction from Naor's lower bound is to first consider deterministic algorithms. Let $\mathcal{A}$ be a deterministic algorithm for approximate MaxIS. Suppose that it takes $T(n)$ rounds in graphs of n nodes. We can use $\mathcal{A}$ to find a maximal independent set in a cycle C of n nodes, as follows. We start by running $\mathcal{A}$ on C to produce an independent set I. Since C is a cycle, there is a natural clockwise ordering for the nodes of I. Between any two consecutive nodes of I, there may be nodes along the cycle that are not adjacent to a node in I. We informally call these nodes the "gaps" between consecutive nodes in I. We can obtain a maximal independent set in C by "filling in" the gap between every two consecutive nodes in I with a maximal independent set (sequentially). To bound the runtime of this algorithm, we need to bound the maximum length of a gap. Since $\mathcal{A}$ is deterministic, it is not very hard to show that the maximum length of a gap is $O(T(n))$. This is because from a local perspective, the nodes cannot distinguish between C and a path of length $\omega(T(n))$, by a standard indistinguishability argument. Hence, one can show that if there is a gap of length $\omega(T(n))$, then $\mathcal{A}$ doesn't return the required approximation on a path of length $\omega(T(n))$. As a result, filling in the gaps between nodes in I takes $O(T(n))$ rounds. Therefore, by running $\mathcal{A}$ on C and then filling in the gaps sequentially, we get an MIS in $O(T(n))$ rounds. And by Linial's lower bound [213], we have that $T(n) = \Omega(\log^* n)$.

However, the argument above fails if $\mathcal{A}$ is a randomized algorithm. The main issue is that when running a randomized algorithm on a cycle, the maximum length of a gap between two consecutive nodes in the independent set can be larger than $O(T(n))$. This is because randomized algorithms that succeed with high probability can fail with probability $1/\text{poly}(n)$, where n is the number of nodes in the graph. Hence, $\mathcal{A}$ can fail on a path of length $O(T(n))$ with probability

$1/\text{poly}(T(n))$ which is non-negligible when $T(n) \ll n$. In particular, since there are $\Omega(|C|/T(n)) = \Omega(n/T(n))$ subpaths of length $O(T(n))$ in C, it is likely that $\mathcal{A}$ fails on at least one of these subpaths. If on the the other hand the number of nodes in the $O(T(n))$-radius neighborhood of a node was larger, then one could hope to get around this issue, as it would amplify the "local" success probability in the neighborhood of a node.

Hence, instead of running $\mathcal{A}$ on C, we run it on a cycle of cliques C_1, which is obtained from C as follows. Each node $v \in C$ is replaced with a clique of size $\approx 2^{|C|}$, denoted by $D(v)$, where every two adjacent cliques are connected by a bi-clique (see also Figure 4.1, for an illustration). By running $\mathcal{A}$ on C_1 instead of C, it boosts the success probability of $\mathcal{A}$ in a small-radius neighborhood of any given node. As a result, a small-radius neighborhood of any node in C_1 must contain a node in the independent set. Using the independent set I_1 that was found in C_1, we can map it to an independent set I in C, as follows. Every $v \in C$ joins I if and only if I_1 contains a node in $D(v)$. Due to the approximation guarantee of $\mathcal{A}$ in C_1, we can prove that the maximum distance between two consecutive nodes in I_1 is small and therefore, the maximum length of a gap in I is small. Finally, we can run a greedy sequential MIS algorithm to fill the gap between every two consecutive nodes in I and find an MIS in C. Hence, if we can find an approximate-MaxIS in C_1 in $o(\log^* |C_1|)$ rounds, then we can find an MIS in C in $o(\log^*(2^{|C|})) = o(\log^* |C|)$ rounds, contradicting Naor's lower bound (Theorem 4.2.2). An illustration of the reduction with all the steps is provided in Figure 4.1 Section 4.6. A detailed reduction is provided in Section 4.6, together with the full proof of the lower bound.

4.3 Preliminaries

Some of our proofs use the following standard probabilistic tools. An excellent source for the following concentration bounds is the book by Alon and Spencer [29]. These bounds can also be found in many lecture notes about basic tail and concentration bounds.

Theorem 4.3.1 (Chernoff Bound). *Let $X = \sum_{i=1}^{n} X_i$, where the X_is are independent random variables with value 0 or 1. Let $\mu = \mathbb{E}[X]$. Then*

1. $\mathbb{P}[X \geqslant (1+\delta)\mu] \leqslant e^{-\frac{\delta^2\mu}{2+\delta}}$ *for all* $\delta > 0$.

2. $\mathbb{P}[X \leqslant (1-\delta)\mu] \leqslant e^{-\mu\delta^2/2}$ *for all* $0 < \delta < 1$.

3. $\mathbb{P}[|X-\mu| \geqslant \delta\mu] \leqslant 2e^{-\mu\delta^2/3}$ *for all* $0 < \delta < 1$.

Theorem 4.3.2. *(**Bernstein's Inequality**). Let* $X_1, ..., X_n$ *be independent random variables such that* $\forall i, X_i \leqslant M$. *Let* X *denote their sum and let* $\mu = \mathrm{E}[X]$ *denote the sum's expected value. Then for any positive* t, *it holds that:*

$$Pr[|X-\mu| \geqslant t] \leqslant 2\exp\left(-\frac{t^2/2}{Mt/3 + \sum_{i=1}^{n} \mathrm{Var}(X_i)}\right)$$

Theorem 4.3.3. *(**One-sided Azuma's Inequality**). Suppose* $\{X_i : i = 0, 1, 2, \ldots\}$ *is a martingale and that* $|X_i - X_{i-1}| \leqslant c_i$ *almost surely. Then, for all positive integers* N *and all positive reals* t,

$$\mathbf{Pr}[X_N - X_0 \leqslant -t] \leqslant \exp\left(-\frac{t^2}{2\sum_{i=1}^{N} c_i^2}\right)$$

Assumptions: In all of our upper and lower bounds in this Chapter, we don't assume that the nodes have any global information. In particular, they don't know n or Δ. The only information that each node has before the algorithm starts is its own identifier, and some polynomial upper bound on n (Since the nodes can send $c\log n$ bits in each round to each of their neighbors, naturally, they know some polynomial upper bound on n).

Some notations: The input graph is denoted by $G_w = (V, E, w)$, where V is the set of nodes, E is the set of edges, and w is the weight function. The reason that we choose to add the weight function in a subscript is that some parts of the analysis deal with graphs that have the same sets of nodes and edges as the input graph, but a different weight function. Hence, such a graph will be denoted by $G_{w'} = (V, E, w')$, to indicate that it is the same as the input graph, but with weight function w' rather than w.

We denote by $N^+(v)$ the inclusive neighborhood of v, which consists of $N(v) \cup \{v\}$, where $N(v)$ is the set of neighbors of v. Furthermore, we denote by $deg(v) = |N(v)|$ the number of neighbors of a node v. Finally, we denote by $w(V')$ the total weight of nodes in $V' \subseteq V$. That is, $w(V') = \sum_{v\in V'} w(v)$.

4.4 Improved Algorithm for General Graphs

In this section we prove Theorems 4.1.1 and 4.1.2.

Theorem 4.1.1 *There is an $O(MIS(n, \Delta)/\epsilon)$-round algorithm in the CONGEST model that finds a $(1+\epsilon)\Delta$-approximation for maximum-weight independent set. Whether the algorithm is deterministic or randomized, depends on the MIS algorithm that is run as a black-box.*

Theorem 4.1.2 *There is a randomized $(\mathrm{poly}(\log\log n)/\epsilon)$-round algorithm in the CONGEST model that finds, with high probability, a $(1+\epsilon)\Delta$-approximation for maximum-weight independent set.*

Theorems 4.1.1 and 4.1.2 share a similar proof structure. First, we present algorithms for $O(\Delta)$-approximation in Sections 5.4 and 4.4.2. Then, by using a general boosting theorem (Theorem 4.4.10 in Section 4.4.3), we get $(1+\epsilon)\Delta$-approximation algorithms.

4.4.1 An $O(MIS(n, \Delta))$-Round Algorithm for $O(\Delta)$-Approximation

In this section we show a very simple $O(MIS(n, \Delta))$-round algorithm that finds an $O(\Delta)$-approximation for MaxIS.

Theorem 4.4.1. *Given a weighted graph $G_w = (V, E, w)$, there is an $O(MIS(n, \Delta))$-round algorithm that finds an independent set of weight at least $\frac{w(V)}{4(\Delta+1)}$, in the CONGEST model. Whether the algorithm is deterministic or randomized depends on the MIS algorithm that is used as a black-box.*

Algorithm For every $v \in V$, let $\delta(v)$ be the maximum degree of a node in the inclusive neighborhood of v. That is, $\delta(v) = \max\{deg(u) \mid u \in N^+(v)\}$. A node v is called *good* if $w(v) \geqslant \frac{1}{2(\delta(v)+1)} \sum_{u \in N^+(v)} w(u)$. The algorithm finds a maximal independent set I in the subgraph induced by the set of good nodes. We prove the following lemma.

Lemma 4.4.2. $w(I) \geqslant w(V)/4(\Delta+1)$

Proof. Let V_{good} be the set of good nodes, and let $\overline{V} = V \setminus V_{good}$. Observe that,

$$\sum_{v\in\overline{V}} w(v) \leqslant \sum_{v\in\overline{V}} \frac{1}{2(\delta(v)+1)} \sum_{u\in N^+(v)} w(u) \leqslant \sum_{v\in V} \frac{deg(v)+1}{2(deg(v)+1)} w(v) = w(V)/2$$

$$\Rightarrow \sum_{v\in I} w(v) \geqslant \sum_{v\in I} \frac{1}{2(\delta(v)+1)} \sum_{u\in N^+(v)} w(u) \geqslant \sum_{v\in I} \frac{1}{2(\Delta+1)} \sum_{u\in N^+(v)\cap V_{good}} w(u)$$

$$\geqslant \frac{1}{2(\Delta+1)} \sum_{v\in V_{good}} w(v) \geqslant w(V)/4(\Delta+1)$$

as desired. Since the value of an optimal solution in G_w is at most $w(V)$, the algorithm returns an $O(\Delta)$-approximation for MaxIS. □

Success with high probability: Given a graph of n nodes, an algorithm that finds a maximal independent set in the graph with high probability is an algorithm that succeeds with probability at least $1 - 1/n^c$ for some constant $c > 1$. In the algorithm above, the black box can be a randomized algorithm that is run on a subgraph $H = (V_H, E_H)$ of G_w. Since $n_H = |V_H|$ is potentially much smaller than n, one may wonder whether the algorithm above actually succeeds with high probability with respect to n. The main idea is to use an algorithm that is *intended* to work for graphs with n nodes, rather than n_H nodes. We prove the following lemma, whose proof is by a simple padding argument.

Lemma 4.4.3. *Let $\mathcal{A}$ be an $MIS(n,\Delta)$-round algorithm that finds a maximal independent set with success probability p in a graph of n nodes and maximum degree Δ. Let $H = (V_H, E_H)$ be a graph of $n_H \leqslant n$ nodes with $(c\log n)$-bit identifiers, for some constant c, and let Δ_H be the maximum degree in H. There is an $O(MIS(n,\Delta_H))$-round algorithm $\mathcal{A}'$ that finds a maximal independent set in H with success probability p.*

Proof. The idea is to pad H with more vertices and then to run an algorithm for maximal independent set on the new graph. In fact, the easiest way to see this is to argue that $\mathcal{A}$ finds a maximal independent set with high probability on the graph H' obtained by adding $n - n_H$ isolated nodes to H with unique identifiers. Since any maximal independent set in H' induces a maximal independent set in H, the claim follows. However, some of the algorithms in the CONGEST model assume that the input graph is connected[3]. To get around the connectivity issue, we define the graph H' obtained by adding a path of $\text{poly}(n)$ nodes with unique $\Theta(\log n)$-bit identifiers to each node that is *local minimum* in H (with respect to the identifiers). Each node that is added to a path connected to a local minimum $u \in V_H$, is given a unique identifier starting with the $c \log n$ bits of the identifier of u as the LSB's (least significant bits), followed by another $c \log n$ bits to ensure that the identifier is unique with respect to the other nodes on the same path. Observe that H' is a graph of $\text{poly}(n)$ nodes, with unique identifiers of $\Theta(\log n)$ bits. Hence, H' is an appropriate input to the CONGEST model. Furthermore, given a maximal independent set I' of H', one can easily find a maximal independent set in H, as follows. Let $I = I' \cap V_H$. Each node that is a local minimum in H joins I if none of its neighbors in H is in I. It holds that I (after adding the additional nodes) is a maximal independent set in H. Since the nodes in H can easily simulate a maximal independent set algorithm in H', without any additional communication cost, it follows that the total running time is $MIS(|V_{H'}|, \Delta_{H'}) + 1$, where $V_{H'}$ and $\Delta_{H'}$ are the set of nodes and maximum degree in H', respectively. Since $\Delta_{H'} \leqslant \Delta_H + 1$, and $|V_{H'}| = \text{poly}(n)$, it holds that $MIS(|V_{H'}|, \Delta_{H'}) = MIS(\text{poly}(n), \Delta_H)$. Moreover, for any n we know that for the specific problem of finding a maximal independent set it holds that $MIS(\text{poly}(n), \Delta) = O(MIS(n, \Delta))$. This is because the round-complexity of finding a maximal independent set is at most logarithmic in the number of nodes. Finally, since a maximal independent set algorithm in H' succeeds with probability $1 - 1/\text{poly}(|V'_H|) \geqslant 1 - 1/\text{poly}(n)$, the claim follows. □

4.4.2 A $\text{poly}(\log\log n)$-Round Algorithm for $O(\Delta)$-Approximation

In this section we show a $\text{poly}(\log \log n)$-round algorithm that finds an $O(\Delta)$-approximation.

[3]This assumption is usually made for global problems such as computing the diameter or all-pairs-shortest-paths. This is because global problems admit an $\Omega(D)$ lower bound, where D is the diameter of the network, which is ∞ for disconnected graphs. While assuming connectivity might not seem reasonable for the MIS problem, for completeness, we want our reduction to hold even for algorithms that make this assumption.

Theorem 4.4.4. *Given a weighted graph $G_w = (V, E, w)$, there is a constant $c > 1$ and a poly$(\log \log n)$-round algorithm in the CONGEST model that finds, with high probability, an independent set of weight at least $\frac{w(V)}{c\Delta}$.*

Our algorithm has the following two-step structure.

1. First, we sample a sparse subgraph $H_w = (V_H, E_H, w)$ of G_w with the following two properties:

   a) The maximum degree Δ_H of H_w is at most $O(\log n)$.

   b) $w(V_H)/\Delta_H = \Omega(w(V)/\Delta)$. That is, the ratio between the total weigh and maximum degree in H_w is at least, up to a constant factor, as in G_w.

2. Then, we use Theorem 4.4.1 to find an independent set in H_w of size at least $\frac{w(V_H)}{4(\Delta_H+1)} = \frac{w(V)}{c\Delta}$, in $O(MIS(n, \Delta_H)) = O(MIS(n, \log n)) = \text{poly}(\log \log n)$ rounds.

The first step: sampling a subgraph with the desired properties. Recall that $w(N(v))$ is the sum of weights of the neighbors of v, which we call the *weighted degree* of v. For each node $v \in V$, let $w_{max}(v) = \max\{w(N(u)) \mid u \in N^+(v)\}$. It is useful to think about $w_{max}(v)$ as the *maximum weighted degree* of a node in the inclusive neighborhood of v. We sample a subgraph $H_w = (V_H, E_H, w)$, as follows. Let $\lambda \geqslant 1$ be a constant to be chosen later. Recall that $\delta(v)$ is the maximum degree of a node in the inclusive neighborhood of v. Each node $v \in V$ joins V_H with probability

$$p(v) = \min\{\lambda \log n \cdot (\frac{1}{\delta(v)} + \frac{w(v)}{w_{max}(v)}), 1\}$$

In Lemma 4.4.5, we show that the maximum degree of H_w is $\Delta_H = O(\log n)$. In Lemma 4.4.9, we show that $w(V_H) = \Omega(\min\{w(V), w(V)\log n/\Delta\})$.

Lemma 4.4.5. *The maximum degree Δ_H in H_w is $O(\log n)$, with high probability.*

Proof. Let $V^+ = \{v \in V \mid p(v) \geqslant 1\}$. We show that each node u has at most $O(\log n)$ neighbors in $V^+ \cap V_H$, and at most $O(\log n)$ neighbors in $(V \setminus V^+) \cap V_H$. Let $N_H(v)$ be the set of neighbors of v in H.

1. For every $v \in V$, $|N_H(v) \cap V^+| \leqslant 2\lambda \log n$: Assume towards a contradiction that there are more than $2\lambda \log n$ nodes in $N_H(v) \cap V^+$. Since each node $v \in V^+$ has $p(v) \geqslant 1$. it holds that
$$\sum_{u \in N(v) \cap V^+} p(u) \geqslant \sum_{u \in N_H(v) \cap V^+} p(u) > 2\lambda \log n$$
On the other hand,
$$\sum_{u \in N(v) \cap V^+} p(u) \leqslant \sum_{u \in N(v)} p(u) = \sum_{u \in N(v)} \lambda \log n \cdot (\frac{1}{\delta(v)} + \frac{w(v)}{w_{max}(v)})$$
Since $deg(v) = |N(v)|$ and $w(N(v)) = \sum_{u \in N(v)} w(u)$ are lower bounds on $\delta(v)$ and $w_{max}(v)$, respectively, we have that
$$\begin{aligned}\sum_{u \in N(v)} \lambda \log n \cdot (\frac{1}{\delta(u)} + \frac{w(u)}{w_{max}(u)}) &\leqslant \sum_{u \in N(v)} \lambda \log n \cdot (\frac{1}{deg(v)} + \frac{w(u)}{w(N(v))}) \\ &= 2\lambda \log n\end{aligned}$$
which is a contradiction.

2. $|N_H(v) \cap (V \setminus V^+)| \leqslant 2\lambda \log n$: Observe that the expected number of neighbors of v in $N_H(v) \cap (V \setminus V^+)$ is
$$\sum_{u \in N(v)} p(u) \leqslant 2\lambda \log n$$
Since $|N_H(v) \cap (V \setminus V^+)|$ is a sum of independent random variables, one can apply Chernoff's bound (Theorem 4.3.1) to achieve that this number concentrates around its expectation with high probability.

By applying a standard Union-Bound argument over all the nodes, we conclude that the maximum degree in H_w is $\Delta_H = O(\log n)$ with high probability. □

The rest of this section is devoted to showing that
$$w(V_H) = \Omega(\min\{w(V), w(V) \log n / \Delta\})$$
This is proved in Lemma 4.4.9. First, we start by proving a slightly weaker lemma, that assumes that for all $v \in V$, $p(v) \leqslant 1$.

Lemma 4.4.6. *Assume $p(v) \leqslant 1$, for all $v \in V$. It holds that $w(V_H) = \Omega(w(V) \log n / \Delta)$, with high probability.*

Main idea of the proof of Lemma 4.4.6: Let $w_1 \geqslant w_2 \geqslant \ldots \geqslant w_n$ be a sorting of the weights of nodes in V in a decreasing order (where ties are broken arbitrarily). Let $V_{high} = \{u \in V \mid w(u) \in \{w_1, \ldots, w_\Delta\}\}$, and let $V_{low} = V \setminus V_{high} = \{u \in V \mid w(u) \in \{w_{\Delta+1}, \ldots, w_n\}\}$. That is, V_{high} contains the Δ heaviest nodes, and V_{low} contains all the other nodes. The proof is split into the following two cases that are proven separately in Claims 4.4.7 and 4.4.8.

1. $w(V_{high}) \geqslant w(V)/2$: In this case, at least half of the total weight is distributed among high-weight nodes. Intuitively, we need to make sure that we get many of these high-weight nodes. Since the number of high-weight nodes that are sampled is a sum of independent random variables, we are able to use Chernoff's bound to prove that many of them are sampled, with high probability. The full proof for this case is presented in Claim 4.4.7.

2. $w(V_{low}) \geqslant w(V)/2$: In this case, at least half of the total weight is distributed among low-weight nodes. Therefore, it is sufficient to show that $w(V_H) = \Omega(w(V_{low}) \log n/\Delta)$. The key property here is that we can bound the maximum weight of a node in V_{low} by $w(V)/\Delta$. We show how to use this property together with Bernstein's inequality to prove Lemma 4.4.6 for this case. The full proof for this case is presented in Claim 4.4.8.

Claim 4.4.7. Assume that for all $v \in V$, $p(v) \leqslant 1$. Let $V_{high} = \{u \in V \mid w(u) \in \{w_1, \ldots, w_\Delta\}\}$. If $w(V_{high}) \geqslant w(V)/2$, then $w(V_H) = \Omega(w(V) \log n/\Delta)$, with high probability.

Proof. Let $S = \{v \in V_{high} \mid w(v) \geqslant w(V)/4\Delta\}$. We start by showing that at least a constant fraction of the total weight in G_w is distributed among nodes in S. Let $\overline{S} = V_{high} \setminus S$, we start by showing that $w(\overline{S}) \leqslant w(V)/4$:

$$w(\overline{S}) \leqslant \sum_{v \in \overline{S}} w(v) \leqslant \sum_{v \in \overline{S}} \frac{w(V)}{4\Delta} \leqslant \frac{w(V)}{4}$$

where the last inequality holds because $|\overline{S}| \leqslant |V_{high}| = \Delta$. Therefore, $w(S) = w(V_{high} \setminus \overline{S}) = w(V_{high}) - w(V(\overline{S})) \geqslant w(V)/4$. Next, we show that $|S \cap V_H| = \Omega(\log n)$, by using Chernoff's bound. Let x_v be a $\{0,1\}$ random variable indicating whether $v \in V_H$, and let $X = \sum_{v \in S} x_v$. We show that the expectation of X is at least $c\log n/4$.

$$\begin{aligned}\mathbb{E}[X] &= \sum_{v\in S}\mathbb{E}[x_v] = \sum_{v\in S}p(v) = \sum_{v\in S}\lambda\log n\cdot(\frac{1}{\delta(v)}+\frac{w(v)}{w_{max}(v)})\\ &\geqslant \sum_{v\in S}\frac{w(v)\lambda\log n}{w(V)} \geqslant \frac{\lambda\log n}{w(V)}\cdot\sum_{v\in S}w(v) = \frac{w(S)\lambda\log n}{w(V)} \geqslant \frac{\lambda\log n}{4}\end{aligned}$$

Furthermore, sine X is a sum of independent $\{0,1\}$ random variables with expectation $\Omega(\log n)$, by applying Chernoff's bound (Theorem 4.3.1), we conclude that there are at least $\Omega(\log n)$ nodes in $S\cap V_H$, with high probability. Since each node in S has weight at least $w(V)/4\Delta$, this implies that the total weight in V_H is $w(V_H)\geqslant w(S\cap V_H)=\Omega(w(V)\log n/\Delta)$, with high probability, as desired. □

Claim 4.4.8. Assume that for all $v\in V$, $p(v)\leqslant 1$. Let $V_{low}=\{v\in V\mid w(v)\in\{w_{\Delta+1},...,w_n\}\}$. If $w(V_{low})\geqslant w(V)/2$, then $w(V_H\cap V_{low})=\Omega(w(V)\log n/\Delta)$, with high probability.

Proof. Let x_v be a $\{0,1\}$ random variable indicating whether $v\in V_H$, let $y_v = x_v\cdot w(v)$, and let $Y=\sum_{v\in V_{low}}y_v$. We prove the following 3 properties:

1. $\mathbb{E}(Y)\geqslant\frac{w(V)\lambda\log n}{2\Delta}$: this is because

$$\begin{aligned}\mathbb{E}[Y] &= \sum_{v\in V_{low}}p(v)\cdot w(v) = \sum_{v\in V_{low}}\lambda\log n\cdot(\frac{1}{\delta(v)}+\frac{w(v)}{w_{max}(v)})\cdot w(v)\\ &\geqslant \sum_{v\in V_{low}}\frac{w(v)\lambda\log n}{\Delta} = \frac{w(V_{low})\lambda\log n}{\Delta} \geqslant \frac{w(V)\lambda\log n}{2\Delta}\end{aligned}$$

   where the last equality holds since $w(V_{low})\geqslant w(V)/2$.

2. For any $v\in V_{low}$, it holds that $w(v)\leqslant w(V)/\Delta$: Recall that $\{w_1,\cdots,w_n\}$ is an ordering of the weight of nodes by a decreasing order. Hence, for any j, it holds that

$$w_j\cdot j\leqslant\sum_{i=1}^{j}w_j\leqslant w(V)$$

   where the first inequality holds because w_j is the minimum among $\{w_1,...,w_j\}$. Hence, since each node $v\in V_{low}$ has weigh w_j where $j>\Delta$, we have that $w(v)\leqslant w(V)/\Delta$ for any $v\in V_{low}$.

3. It holds that $\sum_{v\in V_{low}} \mathbb{E}[y_v^2] \leqslant w(V)\cdot \mathbb{E}[Y]/\Delta$: First, observe that
$$\sum_{v\in V_{low}} \mathbb{E}[y_v^2] \leqslant \max\{w(v) \mid v\in V_{low}\}\cdot \sum_{v\in V_{low}} \mathbb{E}[y_v]$$
$$= \max\{w(v) \mid v\in V_{low}\}\cdot \mathbb{E}[Y] \leqslant \frac{w(V)\cdot \mathbb{E}[Y]}{\Delta}$$
where the last inequality holds by the second property.

By proving these three properties, we have satisfied all the prerequisites of Bernstein's inequality. A direct application of the inequality yields:

$$Pr\big[|Y-\mathbb{E}[Y]| \geqslant \mathbb{E}[Y]/2\big] \leqslant 2\exp\left(-\frac{\mathbb{E}[Y]^2/8}{M\cdot \mathbb{E}[Y]/6+\sum_{v\in V_{low}} \mathrm{Var}(y_v)}\right)$$

By the second and third properties, we have that

$$\sum_{v\in V_{low}} \mathrm{Var}(y_v) = \sum_{v\in V_{low}} \mathbb{E}(y_v^2)-\mathbb{E}[y_v]^2 \leqslant \sum_{v\in V_{low}} \mathbb{E}(y_v^2) \leqslant \frac{w(V)\cdot \mathrm{E}[Y]}{\Delta}$$

$$\Rightarrow Pr\big[|Y-\mathbb{E}[Y]| \geqslant \mathbb{E}[Y]/2\big] \leqslant 2\exp\left(-\frac{\mathbb{E}[Y]^2/8}{\frac{w(V)\cdot\mathbb{E}[Y]}{6\Delta}+\frac{w(V)\cdot\mathbb{E}[Y]}{\Delta}}\right)$$
$$\leqslant 2\exp\left(-\frac{6\Delta\cdot\mathbb{E}[Y]/8}{7w(V)}\right)$$

Furthermore, by the first property, we have that

$$\mathbb{E}[Y] \geqslant w(V)\lambda\log n/2\Delta$$
$$\Rightarrow 2\exp\left(-\frac{6\Delta\cdot\mathbb{E}[Y]/8}{7w(V)}\right) \leqslant 2\exp\left(-\frac{6w(V)\lambda\log n}{56w(V)}\right) = 2\exp\left(-\frac{6\lambda\log n}{56})\right)$$

Finally, choosing $\lambda = 112/6$ implies that:

$$Pr\big[|Y-\mathbb{E}[Y]| \geqslant \mathbb{E}[Y]/2\big] \leqslant \frac{1}{n^{2\log e}} < \frac{1}{n^2}$$

as desired. Furthermore, we can boost the success probability to $1-1/n^k$ for any constant $k>1$, by setting $\lambda = \frac{112k}{3}$.

□

Having proved claims 4.4.7 and 4.4.8, this finishes the proof of Lemma 4.4.6. Lemma 4.4.6 makes the assumption that $p(v) \leqslant 1$ for all $v \in V$. We remove this assumption in the proof of the following lemma.

Lemma 4.4.9. *It holds that $w(V_H) = \Omega(\min\{w(V), w(V)\log n/\Delta\})$, with high probability.*

Proof. Let $V^+ = \{u \in V \mid p(w) \geqslant 1\}$. The proof is split into two cases:

1. $w(V^+) \geqslant w(V)/2$: Since all the nodes in V^+ join V_H deterministically, this implies that $w(V_H) \geqslant w(V^+) \geqslant w(V)/2$.

2. $w(V^+) < w(V)/2$: This implies that $w(V \setminus V^+) \geqslant w(V)/2$. Since each node $w \in V \setminus V^+$ has $p(w) < 1$, we can apply Lemma 4.4.6 directly on the nodes in $V \setminus V^+$ to conclude that $w(V_H) = \Omega(w(V \setminus V^+)\log n/\Delta) = \Omega(w(V)\log n/\Delta)$, with high probability, as desired.

□

Now we are ready to finish the proof of Theorem 4.4.4.

Proof of Theorem 4.4.4. Since both Lemma 4.4.5 and 4.4.9 above hold with high probability, we can apply another standard Union-Bound argument to conclude that both of them hold with high probability (simultaneously). Hence, by running the algorithm from Section 5.4 on H_w, we get an independent set of weight $\Omega(w(V_H)/\Delta_H) = \Omega(w(V)/\Delta)$, in $MIS(n, \Delta_H) = MIS(n, \log n) = \text{poly}(\log\log n)$ rounds, with high probability, as desired. □

4.4.3 Boosting for $(1+\epsilon)\Delta$-Approximation

In this section we prove the following theorem.

Theorem 4.4.10. *Let $\mathcal{A}$ be a T-round algorithm that finds an independent set of weight at least $(\frac{1}{c\Delta})$-fraction of the total weight in the graph, in the CONGEST model. There is a $(\frac{2Tc}{\epsilon})$-round algorithm $\mathcal{A}'$ that finds a $(1+\epsilon)\Delta$-approximation for maximum-weight independent set in the CONGEST model.*

Description of algorithm $\mathcal{A}'$: Our algorithm consists of two stages. In the first stage we iteratively call algorithm $\mathcal{A}$, which returns an independent set, and perform *weight reductions* based on the independent set. Then, we push all nodes in the independent set onto a stack. The next time we invoke $\mathcal{A}$ in the first stage, it is invoked on the graph with the *reduced* weight. In the second stage we iteratively pop the independent sets from the stack and greedily construct another independent set which we return as the solution. $\mathcal{A}'$ is formally given in Algorithm 1. We continue with a formal description of our algorithm.

Let $t = c/\epsilon$. As stated before, there are two stages in $\mathcal{A}'$, each consists of t phases.

First Stage: For $i = 1$ to t phases, in each phase, we first run algorithm $\mathcal{A}$ to find an independent set I_i in G_{w_i}, where $w_1 = w$, and $G_{w_1} = G_w = (V, E, w)$ is the original input graph. Then, we insert the nodes in I_i to a stack S that is initially defined to be empty, and continue to $G_{w_{i+1}} = (V, E, w_{i+1})$, where w_{i+1} is defined as follows. For each $v \in V$,

$$w_{i+1}(v) = \begin{cases} 0 & \text{if } v \in I_i \\ w_i(v) - \sum_{u \in N(v) \cap I_i} w_i(u) & \text{otherwise} \end{cases}$$

That is, w_{i+1} is a weight function which results from weight reductions to w_i, as follows. For each $v \in I_i$, we reduce its total current weight, $w_i(v)$, and therefore its weight becomes zero. For each $v \notin I_i$, we reduce its current weight, $w_i(v)$, by the total weight of its neighboring nodes in I_i. This concludes the first stage. Before we proceed to the second stage, let us define the following weight functions w'_i, for every $i \in [t]$, that are used in the analysis. For each $v \in V$,

$$w'_i(v) = w_i(v) - w_{i+1}(v)$$

Hence, $w'_i(v)$ is the *reduced* weight from v at the end of phase i. We define another weight function w', which is the *total reduced* weight function. For each $v \in V$,

$$w'(v) = \sum_{i=1}^{t} w'_i(v)$$

Second Stage: We construct an independent set I as follows. For $i = 0$ to $t - 1$ phases, we pop out I_{t-i} from the stack, and insert each $v \in I_{t-i}$ to I unless I already contains a neighbor of v.

In Lemma 4.4.13, we prove that I is a $(1+\epsilon)\Delta$-approximation for maximum-weight independent set for G_w. First, let us start with the following helper propositions. Recall that given a weight function $\hat{w}$ we denote by $G_{\hat{w}} = (V, E, \hat{w})$ the same graph as the input graph G_w but with weight function $\hat{w}$ rather than w. For every set of nodes, $V' \subseteq V$, we define $\hat{w}(V') = \sum_{v \in V'} \hat{w}(v)$.

Proposition 4.4.11. *I_i is a Δ-approximation for maximum-weight independent set in $G_{w'_i}$.*

Proof. First, observe that for every $v \in I_i$, it holds that $w'_i(v) = w_i(v) - w_{i+1}(v) = w_i(v)$, and for every $v \notin I_i$, it holds that $w'_i(v) = w'_i(N(v) \cap I_i)$. Let I_i^* be an optimal maximum-weight independent set in $G_{w'_i}$. We can assume that I_i^* contains only nodes in $I_i \cup N(I_i)$, as all the other nodes in $G_{w'_i}$ have zero weight[4]. We have that,

$$w'_i(I_i^*) = w'_i(I_i^* \cap I_i) + w'_i(I_i^* \setminus I_i) = w'_i(I_i^* \cap I_i) + \sum_{v \in I_i^* \setminus I_i} w'_i(N(v) \cap I_i)$$
$$= w'_i(I_i^* \cap I_i) + \Delta w'_i(I_i \setminus I_i^*) \leqslant \Delta w'_i(I_i)$$

as desired. □

In the following proposition we draw a connection between $w(I)$, the final value of our solution, and the total value stored in our stack. Formally, we show the following.

Proposition 4.4.12. *It holds that $w(I) \geqslant \sum_{i=1}^{t} w'_i(I_i) = \sum_{i=1}^{t} w_i(I_i)$.*

Proof. We assume without loss of generality that $\mathcal{A}$ never picks nodes of non-positive weight to the independent set, as we can always remove them and increase the size of the solution. Observe that for every $v \in I$, it holds that $v \in I_i$ for some $i \in [t]$. Let i_v be the phase for which $v \in I_{i_v}$. It holds that,

$$w(v) = w_{i_v}(v) + \sum_{i=1}^{i_v - 1} w_i(N(v) \cap I_i) = w_{i_v}(v) + \sum_{u \in (N(v) \cap (\bigcup_{i=1}^{i_v-1} I_i))} w_{i_u}(u)$$

The first equality is because the weight of v at phase i_v was positive, as otherwise it wouldn't be in I_{i_v}, and, until phase i_v, the total amount of weight that was reduced

[4]Recall that $N(I_i)$ denotes the set of neighbors of all nodes in I_i.

from v is $\sum_{i=1}^{i_v-1} w_i(N(v) \cap I_i)$. And the second equality is because for every u which contributes to the sum $\sum_{i=1}^{i_v-1} w_i(N(v) \cap I_i)$, its contribution is exactly $w_{i_u}(u)$. Hence, we have that,

$$\begin{aligned} w(I) &= \sum_{v \in I} \left(w_{i_v}(v) + \sum_{i=1}^{i_v-1} w_i(N(v) \cap I_i) \right) \\ &= \left(\sum_{v \in I} w_{i_v}(v) \right) + \left(\sum_{v \in I} \sum_{u \in (N(v) \cap (\bigcup_{i=1}^{i_v-1} I_i))} w_{i_u}(u) \right) \\ &\geqslant \left(\sum_{v \in I} w_{i_v}(v) \right) + \left(\sum_{u \in (\bigcup_{i=1}^{t} I_i) \setminus I} w_{i_u}(u) \right) \\ &= \sum_{i=1}^{t} w_i(I_i) \end{aligned}$$

where the last inequality holds because for every $u \in (\cup_{i=1}^{t} I_i) \setminus I$, there is at least one neighbor v of u in I, with $i_v > i_u$. Finally, since for every $i \in [t]$ and for every $v \in I_i$, $w'_i(v) = w_i(v)$, the claim follows. □

Now we are ready to show that I is a $(1+\epsilon)\Delta$-approximation for maximum-weight independent set in the original input graph G_w.

Lemma 4.4.13. *I is a $(1+\epsilon)\Delta$-approximation for maximum-weight independent set in G_w.*

Proof. Let $OPT(G_w)$ be the value of an optimal solution in G_w. The proof is by the following case analysis.

1. $w_t(V) \leqslant \frac{\epsilon}{1+\epsilon} OPT(G_w)$:
   Recall that for all $v \in V$, $w'(v) = \sum_{i=1}^{t} w'_i(v)$. First, observe that,

$$\begin{aligned} w'(V) \geqslant \sum_{v \in V} \sum_{i=1}^{t-1} w'_i(v) &= \sum_{v \in V} \sum_{i=1}^{t-1} w_i(v) - w_{i+1}(v) \\ &= w(V) - w_t(V) \geqslant w(V) - \frac{\epsilon}{1+\epsilon} OPT(G_w) \end{aligned}$$

   Therefore, the value of an optimal solution in $G_{w'}$ cannot be very small compared to the value of an optimal solution in G_w. Namely, $OPT(G_{w'}) \geqslant$

$(1-\frac{\epsilon}{1+\epsilon})OPT(G_w) = OPT(G_w)/(1+\epsilon)$. Finally, by Propositions 4.4.11, I_i is a Δ-approximation to $OPT(G_{w'_i})$, and by Proposition 4.4.12, we have that,

$$w(I) \geqslant \sum_{i=1}^{t} w'_i(I_i) \geqslant \sum_{i=1}^{t} \frac{OPT(G_{w'_i})}{\Delta} \geqslant \frac{OPT(G_{w'})}{\Delta} \geqslant \frac{OPT(G_w)}{(1+\epsilon)\Delta}$$

as desired.

2. $w_t(V) \geqslant \frac{\epsilon}{1+\epsilon}OPT(G_w)$:
Observe that for any $i < t$, it holds that $w_i(V) \geqslant w_t(V)$. Therefore, since $\mathcal{A}$ returns an independent set of weight at least $(\frac{1}{c\Delta})$-fraction of the total weight in the graph, for each phase $i \in [t]$, it holds that $w_i(I_i) \geqslant \frac{\epsilon}{(1+\epsilon)c\Delta}OPT(G_w)$, which implies:

$$\sum_{i=1}^{t} w_i(I_i) \geqslant t\frac{\epsilon}{(1+\epsilon)c\Delta}OPT(G_w) = \frac{1}{(1+\epsilon)\Delta}OPT(G_w)$$

By Proposition 4.4.12, we have that $w(I) \geqslant \sum_{i=1}^{t} w_i(I_i) \geqslant \frac{1}{(1+\epsilon)\Delta}OPT(G_w)$. Which completes the proof.

□

Remark: One can show that the same algorithm also returns an independent set of weight at least $\frac{w(V)}{(1+\epsilon)(\Delta+1)}$. The proof of this argument similar to the proof above. The only difference is that the case analysis in Lemma 4.4.13 is with respect to $\frac{\epsilon}{1+\epsilon}w(V)$. That is, the first case is in which $w_t(V) \leqslant \frac{\epsilon}{1+\epsilon}w(V)$, and the second case is in which $w_t(V) \geqslant \frac{\epsilon}{1+\epsilon}w(V)$. It is straightforward to show that in both cases I is of weight at least $\frac{w(V)}{(1+\epsilon)(\Delta+1)}$, by using the stack property (Proposition 4.4.12).

Corollary 4.4.14. *Let $\mathcal{A}$ be a T-round algorithm that finds an independent set of weight at least $(\frac{1}{c\Delta})$-fraction of the total weight in the graph, in the CONGEST model. There is a $(\frac{2Tc}{\epsilon})$-round algorithm $\mathcal{A}'$ that finds an independent set I of weight at least $\frac{w(V)}{(1+\epsilon)(\Delta+1)}$ in the CONGEST model.*

Algorithm 1 $(1+\epsilon)\Delta$-approx

```
Data: a graph $G = (V, E, w)$
Result: an independent set $I$
1  $I \leftarrow \emptyset$
   $S \leftarrow \emptyset$                                  // $S$ is the Stack
2  $w_1 = w$

3  for $i = 1$ to $t = c/\epsilon$ phases do
4      Run $\mathcal{A}$ on $G_{w_i}$
       Let $I_i \leftarrow \mathcal{A}(G_{w_i})$
       Insert all the nodes in $I_i$ into $S$
       $\forall v \in V : w_{i+1}(v) \leftarrow w_i(v) - \sum_{u \in N^+(v) \cap I_i} w_i(u)$

5  end
6  for $i = 0$ to $t - 1$ phases do
7      Pop $I_{t-i}$ from $S$

8      for $v \in I_{t-i}$ do
9          if $N(v) \cap I = \emptyset$ then
10             add $v$ to $I$
11         end
12     end
13 end
14 return $I$
```

4.5 Even Faster Algorithm for Low-Degree Graphs

In this section we show that there is an $O(1)$-round algorithm that finds an independent set of size $\Omega(n/\Delta)$ for graphs in which $\Delta \leqslant n/\log n$. Using the boosting theorem that is presented in the previous section (in particular, Corollary 4.4.14), this implies that there is an $O(1/\epsilon)$-round algorithm that finds an independent set of size at least $\frac{n}{(1+\epsilon)(\Delta+1)}$ in unweighted graphs of maximum degree $\Delta \leqslant n/\log n$, proving Theorem 4.1.4.

The algorithm is based on a new analysis of one round of Luby's algorithm. This classical algorithm finds an independent set in a graph by independently selecting a rank for each vertex and including a vertex in the output independent set if its rank is greater than the ranks of its neighbors (Algorithm 22).

Algorithm 2 Luby

```
Data: an unweighted graph $G = (V, E)$
Result: an independent set $I$
15 $I \leftarrow \emptyset$
16 for each vertex $u \in V$ do
17     $r_u \leftarrow$ uniformly random number in $\{1, 2, \ldots, 100n^{c+2}\}$
18 end
19 for each vertex $u \in V$ do
20     Add $u$ to $I$ if $r_u > r_v$ for all neighbors $v$ of $u$ in $G$
21 end
22 return $I$
```

In discussion, notice that Luby can be implemented in $O(c)$ rounds in the CONGEST model. We analyze this algorithm by considering a sequential view. The independent set I returned by the algorithm only depends on the order of the r_vs. We could run Luby instead by picking a uniformly random permutation of the vertices and include a vertex v in I if no neighbor of v has a higher rank in the permutation. Furthermore, we can sample the permutation by repeatedly selecting uniformly random vertices without replacement. Equivalently, sample a permutation by repeatedly selecting vertices with replacement, but reject samples seen before (Algorithm 32).

To analyze Luby(G), it suffices to analyze SeqLuby(G):

Proposition 4.5.1. *For any unweighted graph G and constant $c > 0$, SeqLuby(G) produces a distribution over sets I with total variation distance at most $1/n^c$ from the distribution produced by Luby(G); in particular*

$$\sum_{\text{sets } I_0} \left| \Pr_{I \sim \text{SeqLuby}(G)}[I = I_0] - \Pr_{I \sim \text{Luby}(G)}[I = I_0] \right| \leqslant 1/n^c$$

Proof. Let $n = |V|$ and $\mathcal{D}_0$ be the uniform distribution over $\{1, 2, \ldots, 100n^{c+2}\}^n$. This is the distribution over rank tuples $(r_u)_{u \in V}$ used by algorithm Luby. Let $\mathcal{D}_1$ be the uniform distribution over tuples in $\{1, 2, \ldots, n\}^n$ with distinct coordinates. Let Luby$_1$ denote the algorithm with the tuple $(r_u)_{u \in V}$ sampled from $\mathcal{D}_1$ instead of $\mathcal{D}_0$:

Algorithm 3 SeqLuby

```
Data: an unweighted graph $G = (V, E)$
Result: an independent set $I$
$I \leftarrow \emptyset$
$U \leftarrow V$
while $U \neq \emptyset$ do
    $u \leftarrow$ uniformly random element of the set $V$
    $U \leftarrow U \setminus \{u\}$
    if all neighbors $v$ of $u$ are in $U$ then
        Add $u$ to $I$
    end
end
return $S$
```

Algorithm 4 Luby$_1$

```
Data: an unweighted graph $G$
Result: an independent set $I$
$I \leftarrow \emptyset$
$(r_v)_{v \in V} \leftarrow$ sample from $\mathcal{D}_1$
for each vertex $v \in V$ do
    Add $v$ to $I$ if $r_v > r_u$ for all neighbors $u$ of $v$ in $G$
end
return $I$
```

By a union bound over all pairs of vertices, $r_u \neq r_v$ for all $u, v \in V(G)$ with probability at least $1 - \binom{n}{2}\frac{1}{100n^{c+2}} > 1 - 1/(2n^c)$. Let E denote the event $\{r_u \neq r_v \forall u, v \in V(G)\}$ and let $\overline{E}$ denote the negation. Conditioned on $r_u \neq r_v$ for all $u, v \in V(G)$, the output of Luby is identically distributed to the output of Luby$_1$. Therefore,

$$\begin{aligned}
&\sum_{\text{sets } I_0} |\Pr_{I \sim \text{Luby}(G)}[I = I_0] - \Pr_{I \sim \text{Luby}_1(G)}[I = I_0]| \\
&= \sum_{\text{sets } I_0} |\Pr_{I \sim \text{Luby}(G)}[I = I_0] - \Pr_{I \sim \text{Luby}(G)}[I = I_0 | E]| \\
&= \sum_{\text{sets } I_0} |\Pr[I = I_0 | E]\Pr[E] + \Pr[I = I_0 \& \overline{E}] - \Pr[I = I_0 | E]|
\end{aligned}$$

$$\leqslant \sum_{\text{sets } I_0} (\mathbf{Pr}[I = I_0 | E]\, \mathbf{Pr}[\overline{E}] + \mathbf{Pr}[I = I_0 \& \overline{E}])$$
$$= 2\, \mathbf{Pr}[\overline{E}]$$
$$\leqslant 1/n^c$$

so the total variation distance between the output distributions of Luby and Luby_1 is at most $1/n^c$. Next, we show that $\text{Luby}_1(G)$ produces the same distribution over sets as the following algorithm, $\text{SeqLuby}_0(G)$:

Algorithm 5 $\text{SeqLuby}_0(G)$

```
Data: an unweighted graph $G$
Result: an independent set $I$
$I \leftarrow \emptyset$
for $r = n, n-1, \ldots, 1$ do
    $u_r \leftarrow$ uniformly random element of the set $V \setminus \{u_n, u_{n-1}, \ldots, u_{r+1}\}$
    if $u_r$ does not have a neighbor $v$ for which $v = u_s$ for some $s > r$ then
        Add $u_r$ to $I$
    end
end
return $I$
```

Since the u_rs are selected without replacement from V, the distribution over tuples $(u_n, u_{n-1}, \ldots, u_1)$ is a uniform distribution over permutations of V. Let $\mathcal{R} \subseteq \{1, 2, \ldots, n\}^n$ and $\mathcal{U} \subseteq V^n$ denote the families of $\{1, 2, \ldots, n\}$ and V-tuples with distinct coordinates respectively. Fix an ordering $v_1, v_2, \ldots, v_n$ of vertices in G and let $\tau : \mathcal{R} \to \mathcal{U}$ be the map

$$\tau(r_1, r_2, \ldots, r_{n-1}, r_n) = (v_{r_1}, v_{r_2}, \ldots, v_{r_{n-1}}, v_{r_n})$$

τ is a bijection. Furthermore, for any tuple $\mathbf{r} \in \mathcal{R}$, $\text{Luby}_1(G)$ with v_i-rank $\mathbf{r}_i$ outputs the same set S as $\text{SeqLuby}_0(G)$ with vertex ordering $\mathbf{u} = \tau(\mathbf{r})$. Therefore, $\text{Luby}_1(G)$ outputs the same distribution over sets as $\text{SeqLuby}_0(G)$.

Finally, $\text{SeqLuby}_0(G)$ produces the same distribution over independent sets as $\text{SeqLuby}(G)$, because the permutation can be sampled with replacement and rejection of previous samples (as in SeqLuby) rather than without replacement (as in SeqLuby_0). Therefore, $\text{SeqLuby}(G)$ and $\text{Luby}(G)$ produce distributions over sets S with total variation distance at most $1/n^c$, as desired. □

To lower bound the size of the independent set produced by SeqLuby(G), we use an exposure martingale. We start by stopping the SeqLuby algorithm early: we only consider the first $k = n/(2(\Delta+1))$ iterations. Each iteration samples 1 vertex, which precludes at most Δ other vertices from joining the independent set in the future. Therefore, after k iterations, a randomly sampled vertex has a probability of at least $\frac{n-(\Delta+1)k}{n} \geqslant 1/2$ of still being able to join the independent set. Thus, I has size at least $(\frac{1}{2})(\frac{n}{2(\Delta+1)}) = \frac{n}{8(\Delta+1)}$ in expectation. To obtain a high-probability lower bound on the size of I, we use the following proposition, which we prove using Azuma's Inequality:

Proposition 4.5.2. *Consider a set $\mathcal{X}$, a distribution $\mathcal{D}$ over $\mathcal{X}$, a collection $X_1, X_2, \dots, X_k$ of independent, identically distributed random variables sampled from $\mathcal{D}$, and a collection of functions $f_1, f_2, \dots, f_k$, where $f_i : \mathcal{X}^i \to \mathbb{R}$. Suppose that there are numbers $M_0, M_1 > 0$ such that for all $i \in \{1, 2, \dots, k-1\}$ and all tuples $x_1, x_2, \dots, x_i \in \mathcal{X}$, the following conditions hold:*

1. *(Max change) $|f_{i+1}(x_1, x_2, \dots, x) - f_i(x_1, x_2, \dots, x_i)| \leqslant M_0$ and $|f_1(x)| \leqslant M_0$ for all $x \in \mathcal{X}$*
2. *(Expected increase) $\mathbb{E}_{X\sim\mathcal{D}}[f_{i+1}(x_1, x_2, \dots, x_i, X)] \geqslant M_1 + f_i(x_1, x_2, \dots, x_i)$ and $\mathbb{E}_{X\sim\mathcal{D}}[f_1(X)] \geqslant M_1$.*

Then

$$\mathbf{Pr}[f_k(X_1, X_2, \dots, X_k) < kM_1 - t] \leqslant \exp\left(\frac{-t^2}{8M_0^2 k}\right)$$

Proof. Set up a martingale $\{Y_i : i = 0, 1, \dots, k\}$ based on the sequence of function values. Let $Y_0 = 0$ and for all $i \in \{1, 2, \dots, k\}$, let

$$Y_i = f_i(X_1, X_2, \dots, X_i) - \mathbb{E}[f_i(X_1, X_2, \dots, X_i)|X_1, \dots, X_{i-1}] + Y_{i-1}$$

For all integers $i \geqslant 1$, $\mathbb{E}[Y_i|X_1, \dots, X_{i-1}] = Y_{i-1}$, so the Y_is are a martingale. Let f_0 denote the constant function $f_0 = 0$. By the *Max change* condition,

$$|Y_i - Y_{i-1}| = |f_i(X_1, X_2, \dots, X_{i-1}, X_i) - \mathbb{E}[f_i(X_1, X_2, \dots, X_{i-1}, X_i)|X_1, \dots, X_{i-1}]|$$

$$\begin{aligned}
&\leqslant \max_{a,b\in\mathcal{X}} |f_i(X_1, X_2, \ldots, X_{i-1}, a) - f_i(X_1, X_2, \ldots, X_{i-1}, b)| \\
&\leqslant \max_{a,b\in\mathcal{X}} (|f_i(X_1, X_2, \ldots, X_{i-1}, a) - f_{i-1}(X_1, \ldots, X_{i-1})| \\
&+ |f_{i-1}(X_1, \ldots, X_{i-1}) - f_i(X_1, X_2, \ldots, X_{i-1}, b)|) \\
&\leqslant 2M_0
\end{aligned}$$

Therefore, by Theorem 4.3.3,

$$\mathbf{Pr}[Y_k < -t] \leqslant \exp\left(-\frac{t^2}{8kM_0^2}\right)$$

By the *Expected increase* condition, for all integers $i \geqslant 1$,

$$\begin{aligned}
Y_i &= f_i(X_1, X_2, \ldots, X_i) - \mathbb{E}[f_i(X_1, X_2, \ldots, X_i)|X_1, \ldots, X_{i-1}] + Y_{i-1} \\
&= f_i(X_1, X_2, \ldots, X_i) - f_{i-1}(X_1, \ldots, X_{i-1}) \\
&+ f_{i-1}(X_1, \ldots, X_{i-1}) - \mathbb{E}[f_i(X_1, X_2, \ldots, X_i)|X_1, \ldots, X_{i-1}] + Y_{i-1} \\
&\leqslant f_i(X_1, X_2, \ldots, X_i) - f_{i-1}(X_1, \ldots, X_{i-1}) - M_1 + Y_{i-1}
\end{aligned}$$

As a result,

$$Y_k \leqslant f_k(X_1, \ldots, X_k) - kM_1$$

which means that

$$\mathbf{Pr}[f_k(X_1, \ldots, X_k) < kM_1 - t] \leqslant \mathbf{Pr}[Y_k < -t] \leqslant \exp\left(-\frac{t^2}{8kM_0^2}\right)$$

as desired. □

We now prove our main result by letting f_i denote the size of the independent set after i iterations of SeqLuby:

Theorem 4.5.3. *For any $c > 1$, there is an (c)-round CONGEST algorithm Luby(G) that, given a parameter $p \in (0,1)$ and an n-vertex graph G with max degree $\Delta \leqslant n/(256\log(1/p)) - 1$, returns an independent set I for which $|I| \geqslant n/(8(\Delta+1))$ with probability at least $1 - p - 1/n^c$.*

Proof. Luby(G) is an $O(c)$-round algorithm in the CONGEST model. Furthermore, the set I returned is an independent set because each vertex $v \in I$ has a strictly higher rank r_v than its neighbors, which is not simultaneously possible for two adjacent vertices. Therefore, to prove the theorem, we just need to lower bound the size of the set I returned by Luby(G). By Proposition 4.5.1, it suffices to show that the set I returned by SeqLuby(G) has size at least $n/(8(\Delta+1))$ with probability at least $1-p$.

To lower bound the size of I, we apply Proposition 4.5.2 with the following parameter settings:

- $k = n/(2(\Delta+1))$
- $\mathcal{X} = V(G)$
- $\mathcal{D}$: the uniform distribution over $\mathcal{X}$
- X_i: u is sampled during the ith iteration of the while loop in SeqLuby(G).
- $f_i(x_1, x_2, \dots, x_i)$: the function that maps a set of vertices $x_1, \dots, x_i$ to $|I_i|$, where I_i is the set I between the i and $(i+1)$th iterations of the while loop of SeqLuby(G) with u being x_j in the jth while loop iteration.
- $M_0 = 1$
- $M_1 = 1/2$
- $t = k/4$

We now check the conditions of Proposition 4.5.2 with each of these parameters. The *Max change* condition follows immediately from the fact that $I_{i+1} = I_i$ or $I_{i+1} = I_i \cup \{X_{i+1}\}$ for all $i \geqslant 1$ and the fact that $|I_1| \leqslant 1$, so we focus on the *Expected increase* condition. Consider a set of choices $x_1, x_2, \dots, x_i$ of the first i while loop vertices u and let $V_i = \{x_1, x_2, \dots, x_i\}$ for all $i \in \{0, 1, \dots, k\}$. Let X be a random variable denoting the $(i+1)$th vertex u selected by the while loop from U. X is uniformly chosen from V. By the if statement of the SeqLuby algorithm, X is added to I if and only if X is not equal to or adjacent to any vertex in V_i. There are at most $(\Delta+1)|V_i|$ such vertices, so

$$\Pr_{X \sim \mathcal{D}}[f_{i+1}(x_1, x_2, \dots, x_i, X) \neq f_i(x_1, \dots, x_i)] \geqslant 1 - \frac{(\Delta+1)|V_i|}{n}$$

Since $i \leqslant k \leqslant n/(2(\Delta+1))$ and $|V_i| = i$, we get that

$$P_{i,i+1} = \Pr_{X\sim\mathcal{D}}[f_{i+1}(x_1, x_2, \ldots, x_i, X) \neq f_i(x_1, \ldots, x_i)] \geqslant 1/2$$

for all $i \in \{0, 1, \ldots, k-1\}$. Furthermore, we have that

$$\begin{aligned}\mathbb{E}_{X\sim\mathcal{D}}[f_{i+1}(x_1, x_2, \ldots, x_i, X)] &= P_{i,i+1} + f_i(x_1, x_2, \ldots, x_i)\\ &\geqslant 1/2 + f_i(x_1, x_2, \ldots, x_i)\end{aligned}$$

Plugging in our lower bound on the probability shows that the *Expected increase* condition is satisfied. Therefore, Proposition 4.5.2 applies and shows that

$$\mathbf{Pr}[|I_k| < k/4] \leqslant \exp\left(-\frac{(k/4)^2}{8k}\right) = \exp(-k/128)$$

In particular, since $|I| \geqslant |I_k|$,

$$\mathbf{Pr}[|I| < n/(8(\Delta+1))] \leqslant \exp(-n/(256(\Delta+1))) \leqslant p$$

Therefore, the independent set I returned by SeqLuby has the desired size with probability at least $1-p$, as desired. □

4.6 Lower Bound

In this section, we prove Theorem 4.1.3. We do this by a reduction from Naor's [225] lower bound (Theorem 4.2.2). We point out that Naor's lower bound holds even under the assumption that the nodes know the exact value of the number of nodes in the cycle. Our reduction is given in Lemma 4.6.1.

A road-map for this section: This section is organized as follows. We start with Lemma 4.6.1 below, following a description of the reduction. Then, in Section 4.6.1, we point out two important properties of the reduction, which we view as the main intuition behind the proof of Lemma 4.6.1. In Section 4.6.2, we give the formal proof of Lemma 4.6.1. Section 4.6.3 contains the proof of Theorem 4.1.3. Finally, in Section 4.6.4, we improve the probability guarantee in our lower bound, ruling out even algorithms that succeed less often.

Some notations: We say that an algorithm is size-oblivious if it doesn't know the exact value of the number of nodes, but only a polynomial upper bound on it. Similarly, we say that an algorithm is size-conscious if it knows the exact value of the number of nodes.

In Lemma 4.6.1, we show that if there is a size-oblivious algorithm for approximate MaxIS, then there is a size-conscious algorithm that finds a maximal independent set in a cycle.

Lemma 4.6.1. *Suppose that there exists a size-oblivious $T(n)$-round algorithm $\mathcal{A}(G)$ in the LOCAL model that outputs an independent set containing at least $n/(c\Delta)$ nodes in an n-node graph G, with probability at least $1 - p(n)$, where p is a decreasing function. Then, there is a constant $\alpha > 0$ such that for any two integers $n_1 \geqslant n_0$ there is a size-conscious $\alpha T(n_0 n_1)$-round algorithm* RANDMIS(C) *in the LOCAL model that outputs a maximal independent set of an n_0-node cycle graph C with probability at least $1 - n_0 p(n_1)$.*

We now proceed to describing the reduction of Lemma 4.6.1.

The reduction: Let C be a cycle graph of n_0 nodes. The algorithm RANDMIS(C) runs $\mathcal{A}$ on a graph C_1 which is a cycle of cliques, and will be formally defined shortly. After $\mathcal{A}$ finds an independent set in C_1, it uses the resulting independent set to find a maximal independent set in C. We now formally define the graph C_1. For an n_0-node cycle C consisting of vertices $u_1, u_2, \ldots, u_{n_0}$ in that order, let C_1 be a graph on $n_0 n_1$ vertices $\{\{v_{ij}\}_{j=1}^{n_1}\}_{i=1}^{n_0}$. There is an edge between two vertices $v_{ij}, v_{i'j'}$ in C_1 if and only if $|i' - i| \leqslant 1$ or $i' = n_0$ and $i = 1$. The ID of a node v_{ij} in C_1 is the concatenation of the ID for u_i in C and the number j. Notice that these IDs have length at most $\log(n_0 n_1)$. C_1 is a cycle of cliques, with a biclique between two adjacent cliques. This graph is depicted in Figure 4.1.

RANDMIS(C) starts by computing an independent set I_1 in C_1 using $\mathcal{A}(C_1)$, which can be implemented in the LOCAL model on C, with each node u_i simulating all of the actions performed by $\mathcal{A}(C_1)$ on the vertices $\{v_{ij}\}_{j=1}^{n_1}$. Then, the set I_1 is mapped to an independent set I in C, as follows. For each node $u \in C$, it joins I if and only if the corresponding clique in C_1 contains a node in I_1.

Finally, the nodes in I are removed from the cycle together with their neighbors, and RANDMIS(C) finds a maximal independent set in each of the resulting connected components.

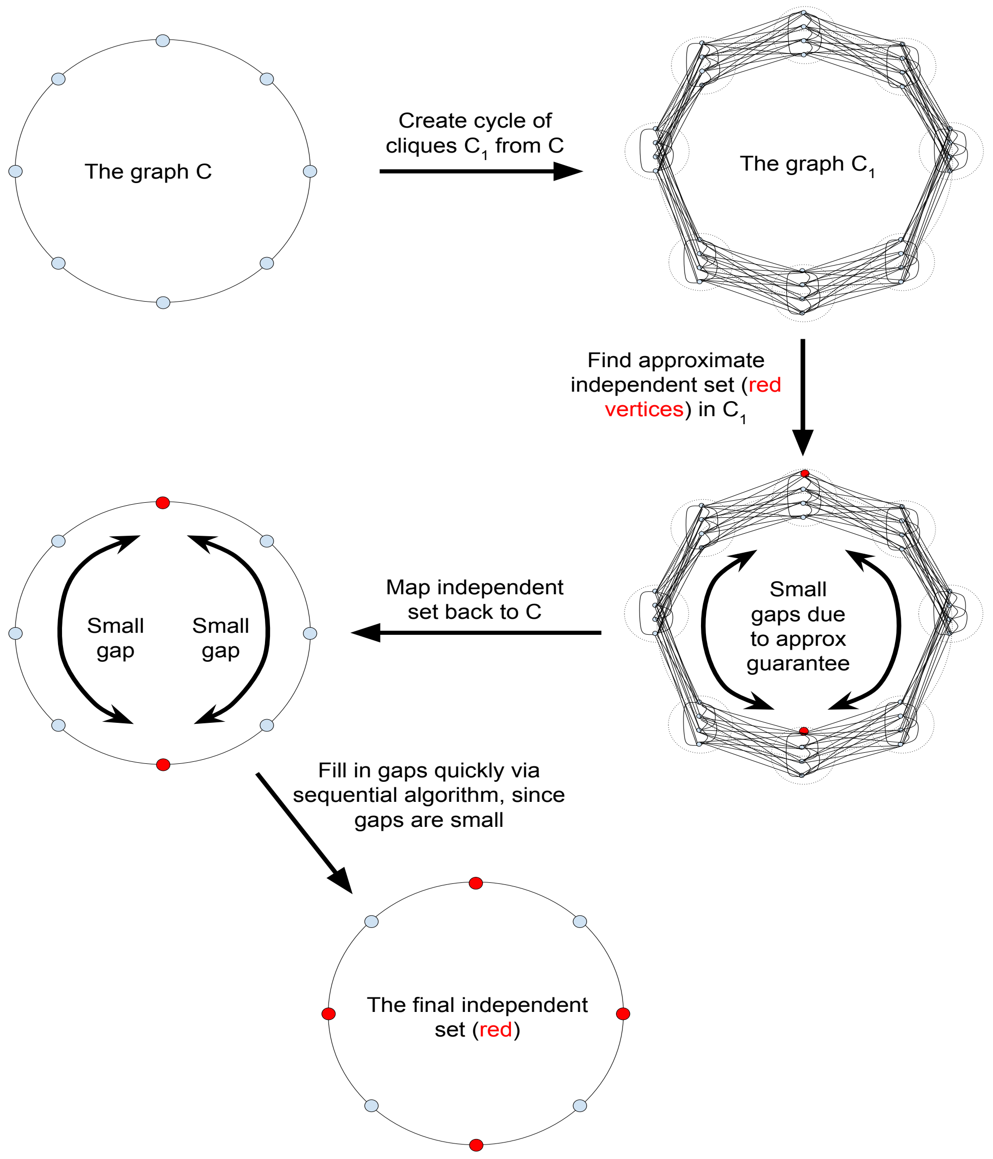
The graph C
Create cycle of cliques C1 from C
The graph C1
Find approximate independent set (red vertices) in C1
Small gaps due to approx guarantee
Map independent set back to C
Small gap
Small gap
Fill in gaps quickly via sequential algorithm, since gaps are small
The final independent set (red)

Figure 4.1: An illustration for our reduction to prove the lower bound. To find a maximal independent set in a cycle C, the nodes run an approximate-MaxIS algorithm to find an independent set I_1 in C_1, which is obtained from C as follows. Each node $v \in C$ is replaced by a large clique of size $2^{|C|}$, denoted by $K(v)$, where every two adjacent cliques are connected by a bi-clique. Using the independent set I_1 in C_1, the nodes map it to find an independent set I in C, as follows. Every $v \in C$ joins I if and only if $K(v)$ contains a node in I_1. Due to the approximation guarantee, the gap between any two nodes in I_1 in C_1 is small, and therefore the gap between any two nodes in I in C is also small. Finally, the nodes run a greedy MIS algorithm to “fill in" the gaps, and find an MIS in C.

Algorithm 6 RANDMIS(C)

```
Data: an $n_0$-node cycle graph $C$
Result: a maximal independent set $S$ of $C$
$S \leftarrow \emptyset$
$I_1 \leftarrow \mathcal{A}(C_1)$ (implemented in LOCAL model on $C$ as stated in Proposition 4.6.8
  proof)                                                              // step (1)

$I \leftarrow \{u_i \in V(C)$ for which there exists $j$ with $v_{ij} \in I_1\}$              // step (2)
Add $I$ to $S$
$J \leftarrow \{u \in V(C) : u \in I$ or $u$ is adjacent to a node in $I\}$
$C_2 \leftarrow C \setminus J$
for each connected component $D$ of $C_2$ in parallel do
    Add a fixed maximal independent set of $D$ to $S$                    // step (3)
end
return $S$
```

4.6.1 Two Key Properties of The Reduction and An Intuitive Explanation

In the proof of Lemma 4.6.1, we crucially exploit two properties of the algorithm $\mathcal{A}$. The first property is important for the correction of our reduction, and the second is important for bounding the run-time of the reduction.

First Property: Global Success. $\mathcal{A}$ is *globally consistent* in the sense that $\mathcal{A}$ returns an *independent set* in the cycle of cliques C_1 with high probability. This property helps us to argue that the returned set for the cycle C is also an independent set. See also Lemma 4.6.2. Observe that the max-degree of C_1 is $3n_1$, which as a function of the number of nodes in C_1 is $\Theta(|V(C_1)|/\log|V(C_1)|)$. This is the reason that in the statement of Theorem 4.1.3 we require the algorithm to be able to succeed on such max-degree graphs.

Second Property: Local Success. The algorithm $\mathcal{A}$ is *locally present* in the sense that an $O(T(n_0n_1))$-neighborhood of any node intersects I_1 with high probability. Since Algorithm $\mathcal{A}$ is size-oblivious, which means that it doesn't have the exact value of the number of nodes in C_1 but only a polynomial approximation of it, it follows that $\mathcal{A}$ doesn't distinguish between the cycle of cliques C_1 and an $O(T(n_0n_1))$-length path of cliques of the same structure. That is, for a node $v \in C_1$, when v runs $\mathcal{A}$ for $T(n_0n_1)$ rounds on C_1, the algorithm behaves exactly the same as if C_1 was a short path of cliques of length $O(T(n_0n_1))$ centered at v, assuming that the ID's of the nodes in the $T(n_0n_1)$-neighborhood of v in the two graphs are the same. We are able to use this observation to show that the $O(T(n_0n_1))$-neighborhood of each node in C_1 must contain a node in I_1 with probability at least $1 - p(n_1)$, as otherwise, the algorithm would fail on a short path of cliques of length $O(T(n_0n_1))$. This in turn implies that the $O(T(n_0n_1))$-neighborhood of any node in the cycle C must contain a node in I with probability at least $1 - p(n_1)$. By a union bound over nodes in C, the distance between any two consecutive nodes in I along the cycle is at most $O(T(n_0n_1))$ with probability at least $1 - n_0p(n_1)$. Therefore, all connected components of $C \setminus I$ have size at most $O(T(n_0n_1))$, so sequentially finding a maximal independent set in each component simultaneously takes $O(T(n_0n_1))$ time to extend I to an MIS for C. See also propositions 4.6.4, 4.6.5, 4.6.7, and 4.6.8.

Later, in the proof of Theorem 4.1.3, we set $T(n) = \log^* n$. Hence, when we say that the algorithm succeeds on a path of cliques H of length $O(T(n_0n_1))$, this graph has maximum-degree $\Theta(|V(H)|/\log^*(|V_H)|)$. This is exactly why the statement of Theorem 4.1.3 requires the algorithm to succeed also on graph with such maximum-degree.

Interestingly, the second property does not hold for one round of Luby's algorithm and the first property does not hold for a $o(\log^* n)$-time greedy algorithm. This perhaps provides another intuitive explanation to why these two properties are manifested to a lower bound proof.

4.6.2 The Proof of Lemma 4.6.1

We start by showing the correctness of the algorithm.

Lemma 4.6.2. RANDMIS(C) *outputs a maximal independent set of the n_0-node cycle graph C with probability at least $1 - p(n_1)$.*

Proof. By definition of the algorithm $\mathcal{A}$, I_1 is an independent set of C_1 with probability at least $1 - p(n_0n_1) \geqslant 1 - p(n_1)$. For the rest of the proof, assume that I_1 is an independent set C_1 (the complement happens with probability at most $p(n_1)$). We now show that I is an independent set. Suppose, instead, that there exist adjacent $u_i, u_{i+1} \in I$. By definition of I, there exist vertices $v_{ij}, v_{(i+1)j'} \in I_1$. By construction of C_1, v_{ij} and $v_{(i+1)j'}$ are adjacent vertices in C_1, a contradiction to the fact that I_1 is an independent set in C_1. Therefore, I must be an independent set in C.

No vertices in C_2 are adjacent to vertices in I within C by definition of C_2. Therefore, S is an independent set in C. Furthermore, each node in J is adjacent to a node in I, while each node in $V(C) \setminus J$ is adjacent to a node in the maximal independent set computed for C_2. Therefore, S is also maximal at the end of the algorithm. Therefore, S is a maximal independent set if I is an independent set, which happens with probability at least $1 - p(n_1)$, as desired. $\square$

Remark 4.6.3. Lemma 4.6.2 holds even if we restrict our attention to algorithms that succeed only on graphs of maximum-degree $\Theta(n/\log n)$. This is because the maximum degree in C_1 is $3n_1$, which as a function of the number of nodes in C_1 is $|V(C_1)|/\log|V(C_1)|$.

The rest of the analysis focuses on the runtime. To bound the runtime, we need to exploit the fact that $\mathcal{A}$ is a size-oblivious distributed algorithm to show that the independent set returned has small gaps with high probability. This is shown by using the fact that $\mathcal{A}$, in the neighborhood of a node v, cannot distinguish between C_1 and an $O(T(n_0n_1))$-length path of cliques containing v. We formalize this in Proposition 4.6.4. Let $R_{\text{large}} = (100c+1)T(n_0n_1) + 2$ and $R_{\text{small}} = 100cT(n_0n_1)$ (recall that c is the constant from the approximation guarantee, where the algorithm returns an IS of size $n/(c\Delta)$). Let L_v denote the set of vertices $u \in V(C_1)$ for which the distance from u to v is at most R_{large}. Let S_v denote the set of vertices $u \in V(C_1)$ for which the distance from u to v is at most R_{small}. Let C_v denote the induced subgraph of C_1 with respect to the set of vertices L_v. Recall that $\mathcal{A}$ is a randomized algorithm that outputs a distribution over sets of vertices in the input graph. We now show the following property of this distribution:

Proposition 4.6.4. *For any node $v \in V(C_1)$, $S_v \cap \mathcal{A}(C_1)$ has the same distribution as $S_v \cap \mathcal{A}(C_v)$.*

Proof. For any node $u \in V(C_1)$, let $f_u(C_1) = \mathbf{1}_{u\in\mathcal{A}(C_1)}$; that is, the indicator function of u's presence in the independent set $\mathcal{A}(C_1)$. Let U_u be the set of vertices in C_1 with distance at most $T(n_0n_1)$ from u. Since $\mathcal{A}$ is a size-oblivious $T(n_0n_1)$-round algorithm in the LOCAL model, f_u is only a function of the randomness, IDs, and edges incident with vertices in U_u for any $u \in V(C_1)$. By definition of f_u,

$$S_v \cap \mathcal{A}(C_1) = \{u \in S_v : f_u(C_1) = 1\}$$

The set $S_v \cap \mathcal{A}(C_1)$ is therefore only a function of the randomness, IDs, and edges incident with vertices in $\cup_{u \in S_v} U_u$. All of this information is contained in the graph C_v, since any node in the set $\cup_{u \in S_v} U_u$ is within distance $R_{\text{small}} + T(n_0 n_1) = R_{\text{large}} - 2$ of v. Therefore, $S_v \cap \mathcal{A}(C_1)$ is only a function of randomness, IDs, and edges in the graph C_v, which means that $S_v \cap \mathcal{A}(C_1)$ is identically distributed to $S_v \cap \mathcal{A}(C_v)$, as desired. □

As a result, to show that S_v contains a node of the independent set with high enough probability, it suffices to think about C_v instead of C_1.

Proposition 4.6.5. *For any node $v \in V(C_1)$, $S_v \cap \mathcal{A}(C_1) \neq \emptyset$ with probability at least $1 - p(n_1)$.*

Proof. By Proposition 4.6.4, $S_v \cap \mathcal{A}(C_v)$ is identically distributed to $S_v \cap \mathcal{A}(C_1)$, so it suffices to lower bound the probability that $S_v \cap \mathcal{A}(C_v)$ is empty. Let $I_v := \mathcal{A}(C_v)$. The maximum degree of vertices in C_v is $3n_1$. Furthermore, $|V(C_v)| = (2R_{\text{large}} + 1)n_1 \geqslant 200cT(n_0 n_1)n_1$. By the output guarantee of $\mathcal{A}$, I_v is an independent set and $|I_v| \geqslant |V(C_v)|/(c(3n_1)) \geqslant 60T(n_0 n_1)$ with probability at least $1 - p(n_1)$. The vertices on $V(C_v) \setminus S_v$ are a union of $2(R_{\text{large}} - R_{\text{small}}) \leqslant 4T(n_0 n_1)$ cliques on n_1 vertices. Therefore, since I_v is an independent set, $|I_v \cap (V(C_v) \setminus S_v)| \leqslant 4T(n_0 n_1)$. Therefore, $|I_v \cap S_v| \geqslant 60T(n_0 n_1) - 4T(n_0 n_1) > 0$ with probability at least $1 - p(n_1)$, as desired. □

Remark 4.6.6. In the proof of Theorem 4.1.3, we set $T(n) = \beta \log^* n$, for a small enough constant β. Under this definition of $T(n)$, Proposition 4.6.5 holds even if we restrict our attention to algorithms that succeed only for graphs of maximum-degree $\Theta(n/\log^* n)$. This is because the maximum-degree in C_v is $3n_1$, which is as a function of the number of nodes in C_v is $\Theta(|C_v|/\log^* |C(v)|)$.

Now, we union bound to prove the desired property for all intervals with width $2R_{\text{small}}$:

Proposition 4.6.7. *Let I be the output of $\mathcal{A}(C_1)$. With probability at least $1 - n_0 p(n_1)$, $S_v \cap I \neq \emptyset$ for all $v \in V(C_1)$.*

Proof. By Proposition 4.6.5 and a union bound over all $i \in \{1, 2, \ldots, n_0\}$, $S_{v_{i1}} \cap I \neq \emptyset$ for all $i \in \{1, 2, \ldots, n_0\}$ with probability at least $1 - n_0 p(n_1)$. For any $j \in \{1, 2, \ldots, n_1\}$, $S_{v_{ij}} = S_{v_{i1}}$. Since every node in C_1 is equal to v_{ij} for some $i \in$

$\{1, 2, \ldots, n_0\}$ and $j \in \{1, 2, \ldots, n_1\}$, $S_v \cap I \neq \emptyset$ for all $v \in V(C_1)$ with probability at least $1 - n_0 p(n_1)$, as desired. □

We now prove a runtime bound:

Proposition 4.6.8. *Given an n_0-node cycle graph C, $\textsc{RandMIS}(C)$ runs in $O(T(n_0 n_1))$ time with probability at least $1 - n_0 p(n_1)$.*

Proof. We go through the RANDMIS algorithm line by line. The call to $\mathcal{A}(C_1)$ can be implemented in the LOCAL model on C as follows. Any T-round LOCAL algorithm can be viewed as independently flipping coins at each node and sending the IDs and randomness of a node u to each node v in its T-neighborhood, followed by no additional communication. This communication can be done in C by having u_i generate the randomness used by all v_{ij}s in $\mathcal{A}(C_1)$. Then, u_i sends this randomness and the IDs of all v_{ij}s to each node in the $T(n_0 n_1)$-neighborhood of u_i in C. Finally, the $\mathcal{A}$ algorithm's execution on v_{ij} can be run on u_i instead. Thus, the call to $\mathcal{A}(C_1)$ takes at most $T(n_0 n_1)$ rounds.

I, J, and C_2 can each be computed in at most two rounds. By Proposition 4.6.7, C_2 has connected components with size at most $O(T(n_0 n_1))$ with probability at least $1 - n_0 p(n_1)$. Thus, for each connected component D of C_2, the vertices $u \in D$ can be sent D in $O(T(n_0 n_1))$ rounds. With no futher communication, the vertices $u \in D$ each use the same algorithm to compute a maximal independent set of D. This completes all lines of the algorithm. Therefore, the algorithm takes $O(T(n_0 n_1))$ time with probability at least $1 - n_0 p(n_1)$, as desired. □

Proof of Lemma 4.6.1. Follows immediately from Lemma 4.6.2 (S is an MIS) and 4.6.8 (for runtime). □

4.6.3 The Proof of Theorem 4.1.3

Theorem 4.1.3 *In the LOCAL model, assuming that the nodes don't know the exact value of n, but only a polynomial upper bound on it, any algorithm that finds an independent set of size $\Omega(n/\Delta)$ in unweighted graphs of n nodes and maximum degree $\Delta \in \{\Theta(n/\log n), \Theta(n/\log^* n)\}$, with success probability $p \geqslant 1 - 1/(10 \log n)$, must spend $\Omega(\log^* n)$ rounds. The lower bound holds even if the nodes know the exact value of Δ.*

Proof. Let α be the constant from Lemma 4.6.1. Suppose, for the sake of contradiction, that there exists a LOCAL algorithm $\mathcal{A}(G)$ that, when given an n-node graph G with maximum-degree $\Delta \in \{\Theta(n/\log n), \Theta(n/\log^* n)\}$, takes at most $T(n) = \frac{1}{100\alpha}(\log^* n)$ time and outputs an $\Omega(n/\Delta)$-node independent set of G with probability at least $1 - 1/(10\log n)$. For some value of n_0, define $n_1 = 2^{n_0}$. By Lemma 4.6.1 there is a $\frac{1}{100}(\log^*(n_0 n_1)) << (1/2(\log^* n_0) - 4)$-round LOCAL algorithm RANDMIS(C) that, given an n_0-node cycle graph C, outputs a maximal independent set of C with probability at least $1 - n_0(1/(10\log(n_1))) = 9/10$. The existence of such an algorithm contradicts Theorem 4.2.2, as desired.

Furthermore, by Remarks 4.6.3 and 4.6.6, it suffices to restrict our attention to graphs of maximum-degree in $\{\Theta(n/\log n), \Theta(n/\log^* n)\}$.

Finally, observe that the max degree in the two hard instances is $3n_1$. Hence, the lower bound holds even if the nodes know the value of the maximum-degree. □

4.6.4 A Note on The Success Probability

In this section we improve the success probability in our lower bound, and show that even algorithms that succeed less often don't exist.

Theorem 4.6.9. *For any constant b, any randomized $o(\log^* n)$-time algorithm that computes an independent set with size greater than $\Omega(n/\Delta)$ in an n-node graph succeeds with probability at most $1 - 1/(10\log^{(b)} n)$, where $\log^{(b)}(x)$ is the function defined recursively as $\log^{(0)}(x) = x$ and $\log^{(b)}(x) = \log\log^{(b-1)}(x)$.*

Proof. The proof is similar to the proof of Theorem 4.1.3. The only difference is that we define n_1 as follows. For some value of n_0, let $n_1^{(0)} = n_0$, $n_1^{(i)} = 2^{n_1^{(i-1)}}$ for all $i > 0$, and $n_1 = n_1^{(b)}$. By Lemma 4.6.1 and the fact that $n_1 \geqslant n_0$, there is an $o(\log^*(n_0 n_1)) = o(b + \log^* n_0) = o(\log^* n_0)$-round LOCAL algorithm RANDMIS(C) that, given an n_0-node cycle graph C, outputs a maximal independent set of C with probability at least $1 - n_0(1/(10\log^{(b)}(n_1))) = 9/10$. The existence of such an algorithm contradicts Theorem 4.2.2, as desired. □

Chapter 5

Distributed Approximate Maximum Matching in Regular Graphs

This chapter is adapter from a joint work with Manish Purohit, Aaron Schild, and Joshua Wang [198], in which we present new upper and lower bounds for approximate Maximum Matching in regular graphs in the distributed model.

5.1 Introduction

The problem of finding a large matching in a graph has garnered significant attention across several central computational models, including the classical sequential model [165,183,218,266,275], dynamic networks [33,76,77,169,172,260], streaming algorithms [36,37,93,149,221,234], online algorithms [92,171,192,194, 222], and distributed computing [16,17,56,71,120,144,147,161,164,174,187,214, 215].

In the classical LOCAL model of distributed computing, there is a network of n nodes that can communicate via synchronized communication rounds. In each round, a node can send an unbounded-size message to each of its neighbors. The goal in the LOCAL model is to solve some task (e.g., find a large matching) while minimizing the number of communication rounds. A simple algorithm by Israeli and Itai [185] finds a maximal matching (MM) in $O(\log n)$ rounds in the LOCAL model with high probability, which constitutes a 2-approximation to maximum matching in unweighted graphs.[1] This matching can be amplified into a $(1+\epsilon)$-

[1]This algorithm uses bounded-size messages, so it works even in the more restricted CON-

approximate matching by incurring a $\text{poly}(1/\epsilon)$ factor in the running time, as shown by Lotker, Patt-Shamir, and Pettie [214].

Later, in a breakthrough, Barenboim, Elkin, Pettie, and Schneider [64] presented a faster algorithm for Maximal Matching in sparse graphs, running in $O(\log \Delta + \text{poly} \log \log n)$ rounds, where Δ is the maximum node degree. Furthermore, Harris [174] showed that the same running time (up to a multiplicative $\text{poly}(1/\epsilon)$ factor) applies for finding a $(1+\epsilon)$-approximate matching with high probability. If one is content with a matching that is only large in expectation, then we can combine recent algorithms for fractional matching, rounding, and amplification to shave a $\log \log \Delta$ factor from the running time [56, 57, 149, 160].

While these algorithms imply a running time of $o(\log n)$ rounds for approximate matching in sparse graphs, achieving a *truly* sub-logarithmic running time ($\log^{1-\delta} n$ for some constant $\delta > 0$) in *general* graphs remains one of the most fascinating mysteries of distributed graph algorithms.[2] On the other hand, the only known lower bound (as a function of n) is $\Omega(\sqrt{\log n / \log \log n})$ rounds, which was shown by Kuhn, Moscibroda, and Wattenhofer [204], and it applies all the way up to a $(\text{poly} \log n)$-approximation.

Regular Graphs: We say that a graph is regular if all the nodes have the same degree. The family of regular graphs has received much attention as a natural benchmark for studying the complexity of various fundamental problems (e.g., [22–27, 40–42, 98, 167, 202, 262, 277]). It is particularly interesting in the context of approximate matching, as regular graphs admit nearly perfect matchings (see for instance [151]) and have been appealing to researchers in various theoretical computer science models [15, 20, 117, 118, 124, 165, 166, 191, 223, 275].

For many problems, the complexity in regular graphs is the same as in general graphs, while for others it becomes significantly lower. Interestingly, as we discuss next, in the LOCAL model, Maximal Matching seems to be of the former kind, while $O(1)$-approximate matching is provably of the latter kind.

In more detail, the hard instances of [204] for approximate matching are very far from being regular. In fact, these instances are also hard for finding an approximate fractional matching, a problem that admits a trivial zero-round solution in regular graphs by simply setting the fractional value for each edge to be $1/\Delta$. This

GEST model, where message size is bounded by $O(\log n)$ bits.

[2]Even only for a $(\text{poly} \log n)$-approximation.

fractional matching can be rounded to an $O(1)$-approximate integral matching by using a simple sampling technique [160].

In contrast, for Maximal Matching, the approximately 40-year-old $O(\log n)$ bound by Israeli and Itai [185] remains the best known even for regular graphs. Moreover, the recent development of the *Round Elimination* technique for regular graphs [47, 49–53, 87–89] has established a lower bound for Maximal Matching of $\Omega(\min\{\Delta, \log\log n/\log\log\log n\})$ rounds for randomized algorithms and $\Omega(\min\{\Delta, \log n/\log\log n\})$ rounds deterministically [49]. For low-degree regular graphs (e.g., cycles), Maximal Matching still requires at least $\Omega(\log^* n)$ rounds [213, 225], even when using randomness.

This raises the question: where does $(1+\epsilon)$-approximate matching fall on the spectrum of round complexity in regular graphs? Is it closer to approximate fractional matching, or does it require some dependence on n similar to Maximal Matching?[3] At first glance, the requirement of finding a matching that is nearly perfect may be at least as challenging as finding a matching that is only maximal (which guarantees only a constant approximation).

In this work, our first result is an algorithm that finds a $(1+\epsilon)$-approximate matching in regular graphs[4] with no dependency on n or Δ. This algorithm implies that $(1+\epsilon)$-approximate matching is strictly easier than Maximal Matching in regular graphs, for an arbitrarily small[5] constant $\epsilon > 0$ (and even for some $o(1)$ values of ϵ).

Theorem 5.1.1. *Let n be a positive integer, and let $\epsilon \in (n^{-1/20}, 1/2)$ be an accuracy parameter. There is an $O(\epsilon^{-5}\log(1/\epsilon))$-round algorithm in the LOCAL model that finds a $(1+\epsilon)$-approximate maximum matching in n-node regular graphs, with high probability. The algorithm works in the CONGEST model for constant values of ϵ.*

[3] Note that the amplification technique of [149] cannot be used to amplify the constant-approximation factor to $(1+\epsilon)$ in regular graphs while using $\text{poly}(1/\epsilon)$ phases of amplification. This is because the technique is designed for general graphs, and applying it to a regular graph can result in a non-regular graph after the first step. Consequently, the amplification algorithm might need to use an algorithm for constant-approximate matching in *general graphs*, which requires some dependence on n or Δ.

[4] All our upper bounds apply also to almost regular graphs, where the degrees of all the nodes are within a $(1 \pm o(1))$-multiplicative factor from each other.

[5] The case where $\epsilon \leqslant n^{-1/20}$ can be handled in $\text{poly}(1/\epsilon)$ rounds by transmitting the entire graph to all nodes.

While this result advances our understanding of $(1+\epsilon)$-approximate matching in regular graphs, a better dependence on ϵ in the runtime is desirable for small values of ϵ. For instance, for $\epsilon \approx 1/\log\Delta$, the algorithm of Harris [174] for general graphs takes $\tilde{O}(\log^4\Delta + \log\log n)$ rounds, while the algorithm from Theorem 5.1.1 takes $\tilde{O}(\log^5\Delta)$ rounds.[6] In our next result, we present an exponentially faster algorithm, provided that $\epsilon > 1/\Delta^c$ for some constant $c > 0$. In other words, we present an exponentially faster $(1+\epsilon)$-approximation algorithm for graphs that are not extremely sparse, i.e., when $\Delta > \text{poly}(1/\epsilon)$. We note that the constant c in Theorem 5.1.2 has not been optimized and can be improved substantially with a more careful analysis.

Theorem 5.1.2 (Main Result)**.** *Let $c = 10^5$ and let $0 < \epsilon < 1$ be a parameter. For Δ-regular graphs with $\Delta > (1/\epsilon)^c$, there is an $O(\log(1/\epsilon))$-round CONGEST algorithm that finds a $(1+\epsilon)$-approximate maximum matching with high probability.*

The restriction in the theorem about the graph being dense enough may sound bizarre: shouldn't we expect the complexity to increase with the density? Indeed, this is the case for nearly all problems in nearly all models of computation. In particular, for our same $(1+\epsilon)$-approximate matching problem in LOCAL in general graphs (not regular ones) the $\approx \log\Delta$ upper bounds by [174] are perfectly consistent with this expectation.[7] Thus, one may think that the restriction for $\Delta > \text{poly}(1/\epsilon)$ in the theorem is a mere technicality that is an artifact of a suboptimal analysis. To our surprise, this is not the case. The following (simple) lower bound shows that our main result of Theorem 1.2 must have benefited from the density of the graph, thus establishing a counter-intuitive separation between sparse and dense regular graphs by which *dense graphs are easier*! Note that, for the same problem, the *opposite* separation holds in general graphs (due to the $\Omega(\min\{\log\Delta/\log\log\Delta, \sqrt{\log n/\log\log n}\})$ lower bound by [204], and the upper bounds by [174]).

Theorem 5.1.3 (Informal version of Theorem 5.7.1)**.** *For any degree $\Delta \geqslant 2$ and error $\epsilon = O(\Delta^{-1})$, any LOCAL algorithm that computes a $(1+\epsilon)$-approximate maximum matching in bipartite Δ-regular graphs with at least $n \geqslant \Omega(\Delta^{-1}\epsilon^{-1})$ nodes requires $\Omega(\Delta^{-1}\epsilon^{-1})$ rounds.*

[6] $\tilde{O}(x)$ hides $\text{poly}\log x$ factors.

[7] Specifically, the upper bounds of [174] are $\tilde{O}(\log\Delta/\epsilon^3 + \text{poly}\log(1/\epsilon, \log\log n))$ and $\tilde{O}(\log^2\Delta/\epsilon^4 + \log^* n/\epsilon)$.

One intuition behind this separation is that in Δ-regular graphs, the number of nearly optimal solutions to maximum matching scales with Δ (see for instance [35]). This abundance of nearly optimal solutions intuitively facilitates the rapid identification of one using randomness.

At the heart of the proof of our main result (Theorem 5.1.2) is a new lemma which we refer to as the *Recursive Regularity Lemma* (Lemma 5.2.2). This lemma shows that running a single round of Luby's algorithm [217] on the line graph of a sufficiently dense Δ-regular graph and removing the matched nodes together with their incident edges yields an almost $\approx \Delta/2$-regular graph.

In Section 5.1.1, we explain how improving our $O(\log(1/\epsilon))$ runtime in Theorem 5.1.2 (even only for very dense graphs where $\Delta \geqslant 2^{\log^{1-\delta} n}$ for some constant $\delta > 0$) might break the $\approx$40-year-old $O(\log n)$-barrier for Maximal Matching in regular graphs. Furthermore, Theorem 5.1.2 implies that regular graphs with $\Delta \geqslant \text{poly}(1/\epsilon)$ are substantially easier than general graphs in the closely related LCA model, which is also discussed in Section 5.1.1.

Outline of this chapter: In the subsequent section, we discuss some implications of our results. Then, in Section 5.1.2 we provide some basic definitions and notation. In Section 5.2 we provide a brief technical overview of our results. Section 5.4 provides a $\text{poly}(1/\epsilon)$-round algorithm for approximate matching in general regular graphs. Section 5.5 and Section 5.6 contain the technical details of our main result (Theorem 5.1.2). Finally, Section 5.7 has the details of our lower bound construction. We defer some basic definitions and concentration inequalities to Section 5.3, as well as a new concentration inequality for sums of random variables using shifted martingale analysis.

5.1.1 Further Implications

We discuss two additional implications of our results. The first is for the related LCA model, and the second is for $(1 - 1/\text{poly}(\Delta))$-approximation in the LOCAL model, an approximation guarantee that is interesting in the context of Maximal Matching, as we discuss below.

The LCA model: A closely related model to distributed graph algorithms is the Local Computation Algorithms (LCA) model, introduced by Alon et al. [28] and Rubinfeld et al. [248]. In the LCA model, we access a graph via adjacency list queries, where each query can ask for the identifier of the i-th neighbor of a node v. The goal of an LCA algorithm is to answer output queries consistently. For instance, in α-approximate matching, an output query involves a node v, and the LCA must determine whether v is in the matching and return the matched neighbor if it is. All the algorithm's answers for all nodes v should be consistent with a single matching, which needs to be α-approximate to the maximum matching. The complexity measure is defined by the maximum number of access queries made by the LCA per output query.

By using a known transformation from the LOCAL model to the LCA model due to Parnas and Ron [230], our main result (Theorem 5.1.2) implies an LCA algorithm for $(1+\epsilon)$-approximate matching with $\Delta^{O(\log(1/\epsilon))}$ queries, in Δ-regular graphs where $\Delta \geqslant \text{poly}(1/\epsilon)$. This stands in sharp contrast with the recent $\Delta^{\Omega(1/\epsilon)}$ lower bound of Behnezhad, Roghani, and Rubinstein [69] for general graphs with maximum degree $\Delta \geqslant \log^4 n$. Observe that our algorithm returns a matching that matches all but an ϵ-fraction of the nodes, which is consistent with the lower bound of [69]. We note that in extremely dense graphs where $\Delta \geqslant n^{1/\log(1/\epsilon)}$, the trivial algorithm of gathering the entire graph has the same query complexity. The following Corollary implies this query complexity for a much wider regime of Δ.

Corollary 5.1.4. *Let $c = 10^5$ and let $0 < \epsilon < 1$ be a parameter. For Δ-regular graphs with $\Delta > (1/\epsilon)^c$, there is an LCA algorithm with $\Delta^{O(\log(1/\epsilon))}$ queries that finds a $(1+\epsilon)$-approximate maximum matching with high probability.*

Nearly perfect matching: It is well-known that any Δ-regular graph contains a matching that matches all but at most a $(1/(\Delta+1))$-fraction of the nodes. Theorems 5.1.1 and 5.1.2 imply that we can get a matching that matches all but a $1/\text{poly}(\Delta)$-fraction of the nodes in $O(\log \Delta)$ rounds in any Δ-regular graph. This is because if Δ is a sufficiently large constant, we can use Theorem 5.1.2 with $\epsilon = 1/\text{poly}(\Delta)$. Otherwise, in sparse graphs where Δ is bounded by a constant, we can use Theorem 5.1.1 with $\epsilon = 1/\Delta$, which is a constant. We note that in general graphs, the best known algorithms for this approximation guarantee take at least $\text{poly}(\Delta)$ rounds [149, 174].

An algorithm that finds such a large matching can be of particular interest to Maximal Matching. Next, we explain how improving our $O(\log \Delta)$ runtime for finding a matching that matches all but a $1/\text{poly}(\Delta)$-fraction of the nodes, even only in very dense graphs where the maximum degree $\Delta \geqslant 2^{\log^{1-\delta} n}$ for some constant $\delta > 0$, could lead to truly sublogarithmic-time algorithm for Maximal Matching in regular graphs. Intuitively, this is because such an algorithm matches all but $\Delta^{1-\gamma}$ neighbors of a node v on average (for some constant $\gamma > 0$). If this algorithm also has a recursive regularity property similar to the one used to prove Theorem 5.1.2 (i.e., if removing the matched nodes and their incident edges results in an approximately $\Delta^{1-\gamma}$-regular graph), we can repeat this algorithm for $\log \log \Delta$ rounds and achieve a sublogarithmic-time algorithm for Maximal Matching. Recently, regular graphs have attracted considerable attention in the context of Maximal Matching and Maximal Independent Set, where lower bounds for regular graphs have been shown using the Round Elimination technique [47, 49–53, 87–89].

5.1.2 Model and Basic Definitions

Basic Graph Notations: For a graph G, we denote by $V(G)$ the set of nodes in G and by $E(G)$ the set of edges. Given a node $u \in V(G)$, we denote by $N_G(u)$ the set of neighbors of u in G, and by $N_G^d(u)$ the set of nodes at distance exactly d from u. When G is clear from the context, we omit the letter G from the notation and use V, E and $N(u)$ for brevity. In Δ-regular graphs, all the nodes have the same degree Δ. In this work, we are interested in unweighted and undirected graphs.

Maximum Matching: A matching $\mathcal{M}$ in a graph G is a set of edges in $E(G)$, where no two edges in $\mathcal{M}$ share a node. A maximum matching in G is a matching of maximum possible size. A $(1+\epsilon)$-approximate matching in G is a matching $\mathcal{M}$ satisfying $OPT \leqslant (1+\epsilon)|\mathcal{M}|$, where OPT is the size of a maximum matching.

5.2 Technical Overview

5.2.1 Warmup: A poly$(1/\epsilon)$-Round Algorithm for General Regular Graphs

To prove Theorem 5.1.1, we use a two stage algorithm. We begin with a Δ-regular graph on n vertices with target error parameter $\epsilon > n^{-1/20}$. The first stage (sampling) involves uniformly and independently sampling edges from the graph with the goal of reducing the degree from Δ to $\mathrm{poly}(1/\epsilon)$, plus some post-processing. After this stage, we have an (irregular) graph on at most n vertices with degree at most $d = \mathrm{poly}(1/\epsilon)$ and have used up a constant fraction of our error parameter ϵ (i.e. this restricted graph still has an almost-perfect matching). Our second stage (matching) involves actually finding a matching in this restricted graph, and its runtime only depends on the error parameter ϵ and the max degree d.

For the sampling stage, uniformly sampling to degree approximately $\epsilon^{-2} \log n$ would result in a graph that retains a near-perfect matching with high probability (via a Chernoff bound plus a union bound). Unfortunately, we do not want the degree to depend whatsoever on n for the sake of the matching stage, so we need to find a way to reduce this degree even further. Instead, we sample down to degree $\Theta(\epsilon^{-4})$. The resulting subgraph can be very irregular, but we manage to tease out enough structure to make our argument go through. In particular, some small fraction of vertices may have degree exceeding our target $\Theta(\epsilon^{-4})$ by more than a factor two. We use Chernoff with bounded dependence and the matching polytope to argue that stripping out these problematic high-degree vertices still leaves an almost-perfect matching.

For the matching stage, we find a matching in the constructed low-degree subgraph from the sampling stage. The state-of-the-art algorithms of Harris [174] fit our task, but they have a small runtime dependence on n. Instead, we combine some of the ideas from [174] with some ideas from [160], as follows. In $\mathrm{poly}(1/\epsilon)$ phases, we increase the size of the matching in each phase by $\mathrm{poly}(\epsilon) \cdot n$ edges. The algorithm for each phase finds a large set of disjoint augmenting paths. This is done by first constructing a hypergraph H with the same set of nodes as in the low-degree subgraph, where each hyperedge corresponds to a $1/\epsilon$-length augmenting paths. Then, we find an $O(1/\epsilon)$-approximate fractional matching in the hypergraph by using the algorithms of [70, 174, 203], and we round this fractional matching by sampling each hyperedge with probability proportional to its fractional value. By using a similar McDiarmid-type argument as in [160], we can show that this rounding produces an integral $O(1/\epsilon)$-approximate matching in the hypergraph H with high probability. Furthermore, observe that the the maximum degree of a node in the hypergraph H is $\exp^{\mathrm{poly}(1/\epsilon)}$. Therefore, the algorithms of [70, 174, 203] for finding a fractional matching in this hypergraph take $O(\mathrm{poly}(1/\epsilon))$ rounds, as desired.

5.2.2 Exponentially Faster Algorithm in Dense Graphs

In this section, we give a brief technical overview for our main result. On bipartite graphs, our algorithm for proving Theorem 5.1.2 simply runs $O(\log(1/\epsilon))$ rounds of Luby's algorithm [217] that finds a large matching in each round. On general graphs, we first run a color coding step to find a large bipartite almost regular subgraph. While the algorithm is very simple, the main challenge is its analysis. In this work, we provide a new martingale-based analysis for Luby's algorithm.

Since our analysis relies on martingale concentration inequalities, it is more convenient to work with the sequential view of Luby's algorithm that was discussed in Chapter 4 Section 4.5. Our key lemma (Lemma 5.2.2) shows that after applying one round of Luby's algorithm (and removing the matched nodes along with incident edges), the remaining graph remains almost regular, i.e., almost all nodes have very similar degrees. Even with the sequential view, classical martingale concentration inequalities are not sufficient to provide the high probability bounds that we require. We use two techniques, a *shifted martingale trick* and *scaled martingale trick*, in order to provide the requisite bounds. Roughly speaking, we show that the number of matched neighbors of a node behaves similarly to a martingale. Finally, we show that a constant fraction of the nodes are matched in each iteration of Luby's algorithm when the graph is almost regular. Combined with our key lemma, this implies that after $O(\log(1/\epsilon))$ rounds at most ϵ fraction of nodes remain unmatched. We now present a deeper overview of each of the above components in the subsections below.

Sequential view of Luby's algorithm

The sequential view of Luby's algorithm was discussed in Chapter 4, Section 4.5, in the context of independent sets. For completeness, in this chapter, we will briefly revisit this view, this time for matching. In the traditional distributed implementation of one round of Luby's algorithm, each edge f picks a uniformly random number r_f and some edge e is chosen into the matching if and only if $r_e < r_{e'}$ for all neighboring edges e'.

We consider the following sequential view - the edges of the graph arrive sequentially in a uniformly random order and an edge e is chosen into the matching if and only if it arrives before any of its neighboring edges.

Assuming that there are no collisions (i.e. each edge chooses a different random number from its neighbors), it can be readily seen that the two algorithms above produce exactly the same distribution over matchings. For CONGEST algorithms, we restrict the range of the random numbers to be integers in $\{1, 2, \ldots, M\}$ for some polynomially large M. In this case, we showed that the two algorithms are identical up to a vanishingly small failure probability in Proposition 4.5.1.

Martingale Techniques

Our analysis of a single round of Luby's algorithm relies on analyzing certain associated martingales and using martingale concentration inequalities.

Shifted Martingale Consider a collection $X_1, X_2, \ldots, X_t$ of boolean random variables and let $\mathbf{E}[X_i \mid X_1, \ldots, X_{i-1}] = p_i$. We are interested in obtaining high probability bounds on the sum $S_t = \sum_{i=1}^{t} X_i$. For example, consider X_i to be the indicator random variable for the event that the ith edge is chosen into the matching. Now clearly, the random variables $\{X_i\}$ are not independent, so we cannot use standard Chernoff bounds. If we let $S_i = \sum_{j=1}^{i} X_j$, then one could hope to use martingale inequalities to get a concentration result for S_i. The challenge here is that the sequence $S_1, \ldots, S_t$ is not necessarily a martingale. Nevertheless, when the p_i are bounded, then we can still utilize martingale concentration inequalities to show that S_t does not deviate too much from its mean by considering the following *shifted random variables*:

$$Y_i = S_i - \mathbb{E}[S_i \mid Y_0, \ldots, Y_{i-1}] + Y_{i-1}$$

Since the sequence $Y_1, \ldots, Y_t$ is a martingale and further has bounded variance, we can use bounded variance martingale concentration bounds on Y_t to give good concentration on S_t as well. This shifting idea was implicitly used in Chapter 4, in the context of approximate-maximum independent set in low-degree graphs. In this chapter, we generalize this shifting idea to broader scenarios in Theorems 5.3.11 and 5.3.12.

Scaled Martingale In some parts of our analysis, the shifted martingale trick doesn't suffice for our purposes. Consider a sequence of random variables denoted by $S_1, \ldots, S_t$ such that $\mathbb{E}[S_i \mid S_1, \ldots, S_{i-1}] = (1-p)S_{i-1}$ for some fixed $0 < p < 1$. In this scenario, we expect the difference $S_i - S_{i-1}$ to decrease as i increases. It is challenging for the shifted martingale trick to exploit such dynamics.

The main reason is that in order for the shifted martingale to give a concentration result, we need the expected value of $S_i - S_{i-1}$ to be bounded by some fixed number, which wouldn't exploit the property that the difference is decreasing over time. To get a concentration result in such scenarios, we use another trick which we refer to as the *scaling trick*. We can obtain a concentration bound for S_t by considering the following scaled random variables:

$$F_i = \frac{S_i}{(1-p)^{i-1}}$$

Observe that F_i is a martingale. This is because $\mathbb{E}[F_i \mid F_1, \cdots, F_{i-1}] = \frac{1}{(1-p)^{i-1}}\mathbb{E}[S_i \mid F_1, \cdots, F_{i-1}] = \frac{1}{(1-p)^{i-1}}\mathbb{E}[S_i \mid S_1, \cdots, S_{i-1}] = \frac{S_{i-1}}{(1-p)^{i-2}} = F_{i-1}$. Therefore, we can get a concentration result for S_t by getting a concentration result for F_t. The main intuition behind the scaling trick is that it exploits the decreasing difference between S_i and S_{i-1} over time. This is exactly the reason for dividing S_i by $(1-p)^i$. For instance, since $F_1 = S_1$, if we get that F_t doesn't deviate too far from F_1, it would imply that $S_i = (1-p)^i F_i \approx (1-p)^i F_1 = (1-p)^i S_1$, which is exactly where we're expecting S_i to be at step i.

Local Recursive Regularity Lemma

We analyze a single round of Luby's algorithm using the above martingale based techniques. First, we show a lemma with the following local guarantee.

Lemma 5.2.1 (Local Recursive Regularity Lemma - Informal)**.** *Let G be a bipartite Δ-regular graph and suppose we run one round of Luby's algorithm on G and let u be an arbitrary node in G. Then, with probability at least $1 - \exp(-\text{poly}(\Delta))$, $\Delta/2 \pm o(\Delta)$ neighbors of u get matched.*

Recall that $N(u)$ is the set of neighbors of node u and $N^2(u)$ is the set of nodes at distance exactly 2 from u. Let A_u be the set of edges between $N(u)$ and $N^2(u)$. To prove Theorem 5.2.1, we show that roughly $\Delta/2$ edges from A_u are chosen into the matching with probability at least $1 - \exp(-\text{poly}(\Delta))$. While the claim holds trivially in expectation, it is challenging to obtain high probability bounds due to the dependencies between u's neighbors.

To simplify exposition, let E_u denote the set of edges that are at most 3 hops away from u, i.e. $E_u = E \cap \{(\{u\} \times N(u)) \cup (N(u) \times N^2(u)) \cup (N^2(u) \times N^3(u))\}$.

Clearly edges not in E_u do not affect any of the edges in A_u and hence can be ignored, so we restrict the analysis to assume that only edges from E_u arrive in the sequential view of Luby's algorithm.

Let $\mathcal{M}_u \subset A_u$ be the set of matching edges chosen from A_u and our goal is to get high probability upper and lower bounds on $|\mathcal{M}_u|$. Let $X_i \in \{0, 1\}$ be an indicator random variable for the event that the ith arriving edge belongs to $\mathcal{M}_u$. Let $q_i = \mathbb{E}[X_i \mid X_1, \ldots, X_{i-1}]$ be the probability that the ith edge is from A_u and none of its neighboring edges have already arrived. One could try now to utilize the shifted martingale trick described above to obtain concentration bounds on $|\mathcal{M}_u| = \sum_i X_i$. However, the main challenge here is that q_i itself is a random variable. Our goal is to analyze q_i using martingales analysis. Roughly speaking, we use the scaled martingale trick discussed above to get concentration results for q_i for all i, which enables us to apply the shifted martingale trick (i.e., Theorems 5.3.11 and 5.3.12) to get the desired bounds on $|\mathcal{M}_u|$.

Analyzing q_i To analyze q_i, we need to understand how many edges *survive* after the first $i-1$ edges have already arrived. Intuitively, an edge is still surviving in iteration i if neither it nor any of its neighbors was not sampled in the first $i-1$ iterations. Let E_i be the set of surviving edges in A_u at the beginning of the ith iteration. Then we have, $q_i = \frac{|E_i|}{|E_u|-(i-1)} = \frac{|E_i|}{\Delta(|N^2(u)|+1)-(i-1)}$ since a sampled surviving edge is always added to the matching.

The final key property towards the proof: To get high probability bounds on $|E_i|$, we first prove that $\mathbb{E}[|E_i| \mid E_{i-1}] \approx (1-2/k)|E_{i-1}|$. This is exactly the setting where the scaled martingale trick can help us to get a concentration result for $|E_i|$, which paves the way for proving Lemma 5.2.1.

Recursive Regularity Lemma

We use Theorem 5.2.1 to show that applying a single round of Luby's algorithm on a regular graph and removing the matched nodes along with incident edges results in a graph that is still almost regular.

Lemma 5.2.2 (Recursive Regularity Lemma - Informal). *Let G be a bipartite Δ-regular graph and let G' be the graph obtained by running one round of Luby's algorithm* *on G and deleting matched nodes and their incident edges. Then all but $o(1)$ fraction of nodes in G' have their degree in the range $\Delta/2 \pm o(\Delta)$ with high probability.*

When Δ is large enough ($\Delta \geqslant \operatorname{poly}\log n$), Theorem 5.2.1 followed by a union bound suffices to show that all nodes have their degree in the range $\Delta/2 \pm o(\Delta)$ with high probability. On the other hand, when Δ is small, then by Theorem 5.2.1, the expected number of nodes that do *not* have the requisite degree is only $n \cdot \exp(-\operatorname{poly}(\Delta))$. Further, the degree of a node after a round of Luby's algorithm only depends on $O(\operatorname{poly}(\Delta))$ other nodes. So, we can use a Chernoff-Hoeffding with bounded dependence inequality to argue that all but an $n \cdot \exp(-\operatorname{poly}(\Delta))$ nodes have the required degree.

Proof Sketch of Theorem 5.1.2

We first argue that running one round of Luby's algorithm on an almost regular graph matches a constant fraction of the nodes with high probability. We note that while this claim is easy to see in expectation, obtaining a high probability bound requires the use of our shifted martingale technique, particularly when the graph becomes only almost regular (instead of fully regular, as in the first iteration). Combined with Theorem 5.2.2 that states that the resulting graph remains almost regular, we get that after $O(\log 1/\epsilon)$ rounds, at most ϵ-fraction of nodes remain unmatched.

5.2.3 Overview of Lower Bounds

To prove lower bounds against LOCAL algorithms, we use some ideas from a lower bound construction of Ben-Basat, Kawarabayashi, and Schwartzman [71] that gave $\Omega(1/\epsilon)$ lower bounds in the LOCAL model for $(1+\epsilon)$ maximum matching as well as other approximate graph problems. The critical idea in that proof is that when an r-round LOCAL algorithm is deciding what to do with a node v (e.g. who to match it with), it can only use the local structure of the graph around v; in particular, it can only see the r-hop neighborhood around v. This means that we could cut out this r-hop neighborhood from the graph, put it back in differently, and the algorithm would have to make the same decision on v (or for randomized algorithms, the same distribution on decisions). The proof revolves around designing these r-hop neighborhoods as (symmetrical) gadgets, then showing that a constant number of gadgets will (with constant probability) induce an unmatched node between them. The BKS proof uses a simple path as their gadget which involves $O(r)$ nodes. Hence the algorithm can be shown to have an overall error rate of one error per $O(r)$ nodes, so the critical round threshold to allow for $(1+\epsilon)$-multiplicative approximations is $\Omega(1/\epsilon)$ rounds.

Relative to their result, the main upgrade we want to make is that the counterexample graph(s) should be Δ-regular. It is relatively straightforward to take BKS path gadgets and stick them into a large cycle, recovering their $\Omega(1/\epsilon)$ round lower bound for 2-regular bipartite graphs. The main technical hurdle we overcome is generalizing to higher degree. We know from our upper bounds that there must be some degradation as the degree Δ increases, so the main question is, how much efficiency do we need to lose to burn our excess degree? We design gadgets with $O(\Delta r)$ nodes and asymptotically maintain the original error rate of one error per constant number of gadgets (the cycle proof argues about the outcome of two gadgets inducing a mistake, but for the general case we reason about the outcome of five gadgets), yielding $\Omega(1/(\Delta\epsilon))$ round lower bounds.

5.3 Preliminaries

5.3.1 Matchings

We use the following well-known characterization of the matching polytope.

Theorem 5.3.1 (Folklore, e.g. [242]). *Let G be an undirected graph and let $\mathcal{M}$ denote the matching polytope for G; that is the convex hull of all 0-1 vectors in $\mathbb{R}^m$ that are indicator vectors of matchings in G. Then, $\mathcal{M}$ can also be written as the intersection of the following families of halfspaces:*

1. $x_e \geqslant 0$ *for all* $e \in E(G)$.
2. $\sum_{e \sim v} x_e \leqslant 1$ *for all* $v \in V(G)$.
3. $\sum_{e \in E(G[S])} x_e \leqslant \frac{|S|-1}{2}$ *for all sets* $S \subseteq V(G)$ *with odd size.*

For a matching M of graph G, an augmenting path P is a path in G that alternates between edges in M and those not in M with the additional property that both its end points are unmatched in M. Let $\mathcal{P}$ be a collection of vertex disjoint augmenting paths, then $M' = M \cup \{\mathcal{P} \setminus M\} \setminus \{M \cap \mathcal{P}\}$ is a new matching of size $|M| + |\mathcal{P}|$ and we say that M' is obtained by *augmenting* M with $\mathcal{P}$.

Proposition 5.3.2 (Theorem 2.1 of [240], statement from Proposition 7.1 of [174])**.** *Let M be an arbitrary matching of G, and OPT be the size of a maximum weight matching in G. For any $\ell \geqslant 1$, there exists a collection $\mathcal{P}$ with $|\mathcal{P}| \geqslant \frac{1}{2}(OPT(1 - 1/\ell) - |M|)$ of vertex disjoint augmenting paths where each path consist of at most $2\ell + 1$ edges.*

5.3.2 Bounded Dependence Concentration Inequality

We utilize the following concentration bound for sums of random variables with limited dependence.

Theorem 5.3.3 (Inequality (3) of [206], derived from Theorem 2.1 of [188])**.** *Consider n random variables $\mathcal{A} = \{X_1, X_2, \ldots, X_n\}$ with the property that $0 \leqslant X_i \leqslant 1$ for all i almost surely. Then*

$$\mathbb{P}\left[\sum_{i=1}^{n} X_i - \sum_{i=1}^{n} \mathbb{E}[X_i] > \lambda\right] \leqslant \exp\left(-\frac{2\lambda^2}{n \cdot \chi(\mathcal{A})}\right)$$

where $\chi(A)$ is the chromatic number of the dependency graph of $\mathcal{A}$.

5.3.3 Martingales

We start with the following useful folklore observations about the conditional expectation of a function of a random variable, and conditional variance.

Observation 5.3.4. Let Y and Z be two random variables such that each is a function of the other. For any random variable X, we have that:

$$\mathbb{E}[X \mid Y] = \mathbb{E}[X \mid Z]$$

Observation 5.3.5. Let X and Y be random variables. It holds that $Var[X \mid Y] = Var[X + f(Y) \mid Y]$ where $f(Y)$ is a function of Y.

Next, we define martingales, supermartingales, and submartingales.

Definition 5.3.6. [Martingale, Supermartingale, and Submartingale]
A sequence of random variables $X_1, \cdots, X_t$ is called a martingale if for any $i \geqslant 2$

$$\mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}] = X_{i-1}$$

Furthermore, the sequence is called a supermartingale if $\mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}] \geqslant X_{i-1}$ for any $i \geqslant 2$, and a submartingale if $\mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}] \leqslant X_{i-1}$ for any $i \geqslant 2$.

We note that in the above definition of super/submartingales, we follow the definition of Chung and Lu [114]. In some other textbooks, the terms are reversed and the condition $\mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}] \geqslant X_{i-1}$ corresponds to a submartingale, and the condition $\mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}] \leqslant X_{i-1}$ corresponds to a supermartingale. We now state some known martingale inequalities.

Theorem 5.3.7. ***(Theorem 6.1 in [114])***
Let $X_1, \cdots, X_t$ be a martingale sequence satisfying:

1. *$Var[X_i \mid X_1, \cdots, X_{i-1}] \leqslant \phi_i$*
2. *$|X_i - X_{i-1}| \leqslant M$*

$\forall i \geqslant 2$, where ϕ_i and M are non-negative constants. Then for $\lambda > 0$,

$$\mathbb{P}[X_t - \mathbb{E}[X_t] \geqslant \lambda] \leqslant \exp\left(-\frac{\lambda^2}{2\left(\left(\sum_{i=1}^{t} \phi_i\right) + M\lambda/3\right)}\right)$$

Theorem 5.3.8. ***(Theorem 6.5 in [114])***
Let $X_1, \cdots, X_t$ be a martingale sequence satisfying:

1. *$Var[X_i \mid X_1, \cdots, X_{i-1}] \leqslant \phi_i$*
2. *$X_{i-1} - X_i \leqslant M$*

$\forall i \geqslant 2$, where ϕ_i and M are non-negative constants. Then for $\lambda > 0$,

$$\mathbb{P}[X_t - \mathbb{E}[X_t] \leqslant -\lambda] \leqslant \exp\left(-\frac{\lambda^2}{2\left(\left(\sum_{i=1}^{t} \phi_i\right) + M\lambda/3\right)}\right)$$

Theorem 5.3.9. ***(Theorem 7.5 in [114]) Bounded Variance Supermartingale***
Let $X_1, \cdots, X_n$ be a supermartingale satisfying:

1. *$Var[X_i \mid X_1, \cdots, X_{i-1}] \leqslant \phi_i$*
2. *$\mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}] - X_i \leqslant M$*

$\forall i \geqslant 2$, where ϕ_i and M are non-negative constants. Then for any $0 < \lambda \leqslant X_1$,

$$\mathbb{P}[X_t \leqslant X_1 - \lambda] \leqslant \exp\left(-\frac{\lambda^2}{2\left(\left(\sum_{i=1}^{t} \phi_i\right) + M\lambda/3\right)}\right)$$

Theorem 5.3.10. ***(Theorem 7.3 in [114]) Bounded Variance Submartingale***
Let $X = X_1, \cdots, X_n$ be a submartingale satisfying:

1. *$Var[X_i \mid X_1, \cdots, X_{i-1}] \leqslant \phi_i$*
2. *$X_i - \mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}] \leqslant M$*

$\forall i \geqslant 2$, where ϕ_i and M are non-negative constants. Then for $\lambda > 0$,

$$\mathbb{P}[X_t \geqslant X_1 + \lambda] \leqslant \exp\left(-\frac{\lambda^2}{2\left(\left(\sum_{i=1}^{t} \phi_i\right) + M\lambda/3\right)}\right)$$

5.3.4 The Shifted Martingale

In this Section we prove the following two theorem by using the shifted martingale trick (that was briefly discussed in Section 5.2.2).

Theorem 5.3.11. ***[The Shifted Martingale Upper Bound]***
For $t > 0$, let $X_1, \cdots, X_t$ be non-negative random variables, $S_i = \sum_{j=1}^{i} X_j$, and $p_i = \mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}]$. Let $P_1, \cdots, P_t$ and M be fixed non-negative numbers, and assume that $X_i \leqslant M$ and $p_i \leqslant P_i$ for all $i \in [t]$. For $P = \sum_{i=1}^{t} P_i$ and $\lambda > P$, it holds that:

$$\mathbb{P}[S_t \geqslant \lambda] \leqslant \exp\left(-\frac{(\lambda - P)^2}{8 \cdot M \cdot P + 2M(\lambda - P)/3}\right)$$

Proof. First, we define the random variable Y_i as follows.

$$Y_i = \begin{cases} 0 & i = 0 \\ S_i - \mathbb{E}[S_i \mid Y_0, \cdots, Y_{i-1}] + Y_{i-1} & i > 0 \end{cases}$$

Roadmap of the proof: Our goal is to show that the sequence $Y_0, \cdots, Y_t$ is a martingale, prove a concentration result for Y_t by using Theorem 5.3.7, and then deduce a concentration result for S_t. To use Theorem 5.3.7, we need to bound the variance $Var[Y_i \mid Y_0, \cdots, Y_{i-1}]$ and the value of $|Y_i - Y_{i-1}|$. The proof is divided into four steps. In the first step, we show that the sequence $Y_1, \cdots, Y_t$ is a martingale. In the second step we show that $Y_i - Y_{i-1} = X_i - p_i$, which implies that $|Y_i - Y_{i-1}| \leqslant 2M$. In the third step, we use the property from the second step to bound the variance $Var[Y_i \mid Y_0, \cdots, Y_{i-1}]$. Finally, in the fourth step, we plug these bounds into Theorem 5.3.7 to get a concentration result for Y_t, and deduce the desired concentration result for S_t.

First step: the sequence $Y_0, \cdots, Y_t$ is a martingale. We show that for any $i \geqslant 1$, $\mathbb{E}[Y_i \mid Y_0, \cdots, Y_{i-1}] = Y_{i-1}$. Observe that:

$$\begin{aligned}\mathbb{E}[Y_i|Y_0, \cdots, Y_{i-1}] &= \mathbb{E}[S_i \mid Y_0, \cdots, Y_{i-1}] - \mathbb{E}[S_i \mid Y_0, \cdots, Y_{i-1}] + \mathbb{E}[Y_{i-1} \mid Y_{i-1}] \\ &= Y_{i-1}\end{aligned}$$

where the second equality follows since for any two random variables X, Y, we have that $\mathbb{E}[\mathbb{E}[X \mid Y] \mid Y] = \mathbb{E}[X \mid Y]$.

Second step: $Y_i - Y_{i-1} = X_i - p_i$. The claim trivially holds for $i = 1$. For $i > 1$, observe that:

$$\begin{aligned}Y_i - Y_{i-1} &= S_i - \mathbb{E}[S_{i-1} \mid Y_0, \cdots, Y_{i-1}] && (1)\\ &= S_i - \mathbb{E}[S_i - S_{i-1} + S_{i-1}] \mid Y_0, \cdots, Y_{i-1}] && (2)\\ &= S_i - \mathbb{E}[S_i - S_{i-1} \mid Y_0, \cdots, Y_{i-1}] - \mathbb{E}[S_{i-1} \mid Y_0, \cdots, Y_{i-1}] && (3)\\ &= S_i - \mathbb{E}[S_i - S_{i-1} \mid S_1, \cdots, S_{i-1}] - \mathbb{E}[S_{i-1} \mid S_1, \cdots, S_{i-1}] && (4)\\ &= S_i - S_{i-1} - \mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}] && (5)\\ &= X_i - \mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}] && (6)\\ &= X_i - p_i && (7)\end{aligned}$$

where (3) follows from linearity of expectation, (4) follows from Observation 5.3.4 (5) follows since $\mathbb{E}[S_{i-1} \mid S_1, \cdots, S_{i-1}] = \mathbb{E}[S_{i-1} \mid S_{i-1}] = S_{i-1}$, and from another application of Observation 5.3.4 since $(X_1, \cdots, X_{i-1})$ and $(S_1, \cdots, S_{i-1})$ are functions of each other, so conditioning on either of them is equivalent.

Third step: $Var[Y_i \mid Y_0, \cdots, Y_{i-1}] \leqslant 4MP_i$. Observe that

$$Var[Y_i \mid Y_0, \cdots, Y_{i-1}] = Var[Y_i - Y_{i-1} \mid Y_0, \cdots, Y_{i-1}] \qquad (1)$$

$$\begin{aligned}
&\leqslant \mathbb{E}[(Y_i - Y_{i-1})^2 \mid Y_0, \cdots, Y_{i-1}] && (2)\\
&\leqslant 2M \cdot \mathbb{E}[|X_i - p_i| \mid Y_0, \cdots, Y_{i-1}] && (3)\\
&\leqslant 2M \cdot (\mathbb{E}[X_i \mid Y_0, \cdots, Y_{i-1}] + \mathbb{E}[p_i \mid Y_0, \cdots, Y_{i-1}]) && (4)\\
&= 2M \cdot (\mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}] + \mathbb{E}[p_i \mid X_1, \cdots, X_{i-1}]) && (5)\\
&\leqslant 2M(p_i + \mathbb{E}[p_i \mid X_1, \cdots, X_{i-1}]) && (6)\\
&\leqslant 4MP_i && (7)
\end{aligned}$$

where (1) follows from Observation 5.3.5, (2) follows from the definition of variance, (3) follows since we showed that $Y_i - Y_{i-1} = X_i - p_i$ in the second step, which also implies that $|Y_i - Y_{i-1}| \leqslant |X_i| + |p_i| \leqslant 2M$, (4) follows from the triangle inequality, linearity of expectation, and since the X_i's are non-negative, (5) follows from Observation 5.3.4, and (7) follows since $p_i \leqslant P_i$ for all i.

Fourth step: concentration of Y_t and S_t. First, observe that $Y_0 = \mathbb{E}[Y_t] = 0$. Moreover, having proved that the sequence $Y_1, \cdots, Y_t$ is a martingale, $|Y_i - Y_{i-1}| \leqslant 2M$, and $Var[Y_i \mid Y_0, \cdots, Y_{i-1}] \leqslant 4MP_i$, we can plug these bounds into Theorem 5.3.7 to that for $\lambda' > 0$ and $P = \sum_{i=1}^{t} P_i$:

$$\mathbb{P}[Y_t \geqslant \lambda'] \leqslant \exp\left(-\frac{\lambda'^2}{8 \cdot M \cdot P + 2M \cdot \lambda'/3}\right)$$

Furthermore, since $Y_i - Y_{i-1} = X_i - p_i$ for all $i \in [t]$, it implies that $Y_t = S_t - \sum_{i=1}^{t} p_i$. Hence, $Y_t \geqslant S_t - P$, which implies that for $\lambda > P$:

$$\mathbb{P}[S_t \geqslant \lambda] \leqslant \mathbb{P}[Y_t + P \geqslant \lambda] = \mathbb{P}[Y_t \geqslant \lambda - P] \leqslant \exp\left(-\frac{(\lambda - P)^2}{8M \cdot P + 2M(\lambda - P)/3}\right)$$

as desired. □

Theorem 5.3.12. ***[The Shifted Martingale Lower Bound]***
For $t > 0$, let $X_1, \cdots, X_t$ be non-negative random variables, $S_i = \sum_{j=1}^{i} X_j$, and $p_i = \mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}]$. Let $P_1^\ell, \cdots, P_t^\ell, P_1^h, \cdots, P_t^h$ and M be fixed non-negative numbers, and assume that $X_i \leqslant M$ and $P_i^\ell \leqslant p_i \leqslant P_i^h$ for all $i \in [t]$. Let $P^\ell = \sum_{i=1}^{t} P_i^\ell$ and $P^h = \sum_{i=1}^{t} P_i^h$. For $P^\ell > \lambda$, it holds that:

$$\mathbb{P}[S_t \leqslant \lambda] \leqslant \exp\left(-\frac{(P^\ell - \lambda)^2}{8 \cdot M \cdot P^h + 2M \cdot (P^\ell - \lambda)/3}\right)$$

Proof. The proof is very similar to the proof of Theorem 5.3.11. We start with defining the shifted random variable Y_i similarly to how we defined it in the proof of Theorem 5.3.11.

$$Y_i = \begin{cases} 0 & i = 0 \\ S_i - \mathbb{E}[S_i \mid Y_0, \cdots, Y_{i-1}] + Y_{i-1} & i > 0 \end{cases}$$

We showed in the proof of Theorem 5.3.11 that the sequence $Y_1, \cdots, Y_t$ is a martingale. Next, we would like to use Theorem 5.3.8 to get a concentration result for Y_t, which would imply a concentration result for S_t. Recall that in the proof of Theorem 5.3.11 we showed that $Var[Y_i \mid Y_0 \cdots, Y_{i-1}] \leqslant 4MP_i^h$ (which was shown in the third step in the proof of Theorem 5.3.11), and that $Y_t = S_t - \sum_{i=1}^t p_i$ (which was shown at the end of the proof of Theorem 5.3.11). Furthermore, recall that $\mathbb{E}[Y_t] = 0$. Hence, we can plug these bounds into Theorem 5.3.8 to get that:

$$\begin{aligned}\mathbb{P}[S_t \leqslant \lambda] \leqslant \mathbb{P}[Y_t + P^\ell \leqslant \lambda] &= \mathbb{P}[Y_t \leqslant -(P^\ell - \lambda)] \\ &\leqslant \exp\left(-\frac{(P^\ell - \lambda)^2}{8 \cdot M \cdot P^h + 2M \cdot (P^\ell - \lambda)/3}\right)\end{aligned}$$

as desired. □

5.4 Warmup: A poly$(1/\epsilon)$-Round Algorithm

In this we show a simple randomized algorithm that finds a $(1+\epsilon)$-approximate matching in a number of rounds which only depends on the accuracy ϵ and not the graph size or degree.

Theorem 5.1.1. *Let n be a positive integer, and let $\epsilon \in (n^{-1/20}, 1/2)$ be an accuracy parameter. There is an $O(\epsilon^{-5}\log(1/\epsilon))$-round algorithm in the LOCAL model that finds a $(1+\epsilon)$-approximate maximum matching in n-node regular graphs, with high probability. The algorithm works in the CONGEST model for constant values of ϵ.*

While the runtime in Theorem 5.1.1 is constant for constant ϵ, the dependence on ϵ is rather high. We can obtain better dependencies on ϵ by leveraging the $(1+\epsilon)$-approximate matching algorithms given by Harris [174] at the cost of small dependencies on n. This may be preferable for certain pairs of ϵ and n.

Our algorithm for proving this result runs in two stages. In the first stage, we reduce the degree Δ to $\text{poly}(1/\epsilon)$ by sampling edges independently with probability $\Theta(1/(\Delta\epsilon^4))$ and then only consider the subgraph induced by nodes whose degree is approximately $\Theta(1/\epsilon^4)$. In Theorem 5.4.1, we show that the resulting subgraph retains a large matching with high probability.

In the second stage, we need to find an almost perfect matching in the sampled graph. For the second stage of Theorem 5.1.1, we design a novel algorithm without any dependencies on n which finds an almost perfect matching in the sampled graph. Our algorithm runs in rounds; each round extends an existing matching by finding augmenting paths of limited length.

We now handle each stage in separate subsections: Section 5.4.1 for the sampling stage and Section 5.4.2 for the matching stage.

5.4.1 Sampling Stage

In this subsection, we give our sampling stage algorithm and prove that the resulting graph retains an almost perfect matching. Algorithm 7 gives a formal description of the algorithm.

Algorithm 7 SamplingStage(G)

```
Input: A $\Delta$-regular unweighted graph $G = (V, E)$; an accuracy parameter $\epsilon \in (n^{-1/20}, 1/2)$
Output: An unweighted graph $G' = (V', E')$ and a max degree $d$ which is $\min\{\frac{6000}{\epsilon^4}, \Delta\}$ with the guarantee the max degree of $G'$ is at most $d$
58 if $\Delta \leqslant \frac{6000}{\epsilon^4}$ then
59     Return $G' \leftarrow G$ and max degree $d \leftarrow \Delta$.
60 end
61 $E_1 \subseteq E \leftarrow$ each edge in $E$ is included independently with probability $p' = \frac{3000}{\Delta\epsilon^4}$
62 $G_1 \leftarrow (V, E_1)$
63 $V_2 \leftarrow$ all vertices in $V$ with degree at most $2p'\Delta$ in $G_1$
64 $G_2 \leftarrow G_1[V_2]$
65 Return graph $G' \leftarrow G_2$ and max degree $d \leftarrow (2p'\Delta)$.
```

The main goal of this subsection is to prove the following result about Algorithm 7.

Lemma 5.4.1. *Let $OPT(H)$ denote the size of the maximum matching in graph H. Then, when we run Algorithm 7, we have that $OPT(G_2) \geqslant (1-5\epsilon)OPT(G)$ with probability at least $1-5/n^{30}$.*

We observe that if $\Delta \leqslant \frac{6000}{\epsilon^4}$, then Theorem 5.4.1 follows trivially (keeping the entire graph means we preserved the original matching). Thus, in the rest of this section, we assume that we used the sampling probability $p' = \frac{3000}{\Delta\epsilon^4} < 1$. Our plan is to show that the intermediate graph G_1 has an almost-perfect matching and that there are not many high-degree vertices so removing them does not significantly reduce the size of the matching.

More formally, we plan to invoke the following folklore result about almost-regular graphs admitting an almost-perfect matching. The result requires the following notation. Let $\kappa \in (0,1/2)$ and $D \geqslant 1$ be fixed. We say an edge e is (κ, D)-*balanced* if and only if both its endpoints have degree in $((1-\kappa)D, (1+\kappa)D)$.

Lemma 5.4.2 (Folklore). *Let $G = (V, E)$ be an undirected, unweighted graph and $\tau_e, \tau_v, \kappa \in (0, 1/2)$ and $D \geqslant 1$ be fixed such that -*

(a) At least $(1-\tau_e)|E|$ edges are (κ, D)-balanced.

(b) At least $(1-\tau_v)|V|$ vertices have degree in $((1-\kappa)D, (1+\kappa)D)$.

Then G has a matching of size at least $(1-\tau_e-\tau_v-2\kappa-\frac{1}{D+1})\frac{|V|}{2}$ consisting only of (κ, D)-balanced edges.

Proof. Let X be the set of vertices in G with degrees in $((1-\kappa)D, (1+\kappa)D)$. Restricting the graph to these vertices yields $H = G[X]$. Then by definition, we have $|V(H)| \geqslant (1-\tau_v)|V(G)|$ and $|E(H)| \geqslant (1-\tau_e)|E(G)|$. Let $D = (1+\kappa)D$. Our goal is to find a large matching in H, which we plan to do by providing a fractional point in the matching polytope and deducing that some integral point (matching) is at least as good.

Our fractional point is $x \in \mathbb{R}^{E(H)}$ where $x_e = \frac{1}{D+1}$, $\forall e \in E(H)$. We first claim that x belongs to the matching polytope of H by verifying that it satisfies the conditions of Theorem 5.3.1. The first set of inequalities is true as $x_e = \frac{1}{D+1} > 0$. Since the degree of any vertex in H is at most D by definition, the second set of inequalities is satisfied as well. For the third set of inequalities, let $S \subseteq X$ be an arbitrary odd sized set of vertices. Note that $|E(H[S])| \leqslant \frac{|S|(|S|-1)}{2}$ and further since each vertex in H has degree at most D, $|E(H[S])| \leqslant \frac{1}{2}\sum_{v\in S} D = \frac{|S|D}{2}$. Thus we have,

$$\begin{aligned}
\sum_{e\in E(H[S])} x_e = |E(H[S])| \cdot \frac{1}{D+1} &\leqslant \frac{|S|\min(|S|-1,D)}{2(D+1)} \\
&= \frac{|S|\min(|S|-1,D)+(D+1)}{2(D+1)} - \frac{1}{2} \\
&\leqslant \frac{\max(|S|,D+1)\min(|S|-1,D)+\max(|S|,D+1)}{2(D+1)} - \frac{1}{2} \\
&= \frac{\max(|S|,D+1)\min(|S|,D+1)}{2(D_0+1)} - \frac{1}{2} \\
&= \frac{|S|(D+1)}{2(D+1)} - \frac{1}{2} = \frac{|S|-1}{2}
\end{aligned}$$

and thus the third set of inequalities is also satisfied. Thus, by Theorem 5.3.1, x is in the convex hull of indicator vectors of matchings in H, which implies that there exists an integral matching M in H with size at least $\sum_{e\in E(H)} x_e$. In particular,

$$|M| \geqslant \sum_{e\in E(H)} x_e = \frac{|E(H)|}{D+1} \geqslant \frac{(1-\tau_e)|E(G)|}{D+1}$$

But we have $|E(G)| \geqslant \frac{1}{2}\cdot(\sum_{v\in V(H)}(1-\kappa)D) \geqslant \frac{1}{2}\cdot((1-\tau_v)|V(G)|(1-\kappa)D)$

$$\begin{aligned}
&\geqslant \frac{(1-\tau_e)(1-\tau_v)(1-\kappa)|V(G)|D}{2(D+1)} \geqslant \frac{(1-\tau_e)(1-\tau_v)(1-\kappa)|V(G)|D}{2(1+\kappa)(D+1)} \\
&\geqslant (1-\tau_e)(1-\tau_v)(1-2\kappa)(1-1/(D+1))\frac{|V(G)|}{2} \\
&\geqslant (1-\tau_e-\tau_v-2\kappa-1/(D+1))\frac{|V(G)|}{2}
\end{aligned}$$

as desired. □

Now we just need to show that we can make the subgraph G_1 sampled in Algorithm 7 fit the conditions of Theorem 5.4.2. We expect vertices to have degree $p'\Delta$. Since we are going to need κ to be on the order of ϵ, we are aiming to have a lot of vertices with degree in the range $((1-\epsilon)p'\Delta, (1+\epsilon)p'\Delta)$. Unfortunately, as each edge is sampled independently with probability $p' = \Theta(\frac{1}{\Delta\epsilon^4})$, a standard Chernoff bound followed by a union bound attempting to guarantee that all vertices have degree between $((1-\epsilon)p'\Delta, (1+\epsilon)p'\Delta)$ results in too much failure probability. Instead, for the sake of analysis, we sample the edges in two separate phases and utilize concentration bounds for random variables with limited dependence to show that most of the vertices have approximately the correct degree. Formally, we show the following lemma.

Lemma 5.4.3. *With probability at least $1 - 4/n^{30}$, both of the following conclusions hold:*

(a) At most $(4e^{-1500/(27\epsilon^2)})n$ vertices have degree outside of $((1-\epsilon)p'\Delta, (1+\epsilon)p'\Delta)$ in G_1.

(b) At most $(12e^{-1500/(27\epsilon^2)})p'm$ edges in G_1 are not $(\epsilon, p'\Delta)$-balanced.

In order to prove Theorem 5.4.3, we break up the sampling stage into two phases as follows. Let $p = \min\{\frac{1000\log n}{\Delta\epsilon^2}, 1\}$ and $q = p'/p$. For the sake of analysis, we assume that the set E_1 in Algorithm 7 is constructed as follows: let $E_0 \subseteq E$ be such that each edge e is included in E_0 independently with probability p, and let $E_1 \subseteq E_0$ be such that each edge is chosen independently with probability q. Let $G_0 = (V, E_0)$. The following two propositions are simple consequences of Chernoff bounds.

Proposition 5.4.4. *With probability at least $1 - 1/n^{30}$, every vertex in G_0 has degree between $(1-\epsilon/3)p\Delta$ and $(1+\epsilon/3)p\Delta$.*

Proof. Note that if $p = 1$, then the claim follows trivially. So suppose $p = \frac{1000\log n}{\Delta\epsilon^2} < 1$. Consider a vertex $v \in V$. For each neighbor u of v in G, let X_{uv} denote the indicator variable of the presence of the edge $\{u, v\}$ in G_0. Let $X_v = \sum_{u\in N_G(v)} X_{uv}$ be the degree of v in G_0. Note that $\mathbb{E}[X_v] = \sum_{u\in N_G(v)} \mathbb{E}[X_{uv}] = \sum_{u\in N_G(v)} p = p\Delta$. Thus, by Theorem 4.3.1 with $\delta \leftarrow \epsilon/3$,

$$\mathbb{P}[|X_v - p\Delta| \geqslant \epsilon/3p\Delta] \leqslant 2e^{-\epsilon^2 p\Delta/27} \leqslant 2/n^{1000/27}$$

Thus, by a union bound over all $v \in V(G)$, the probability that there exists a vertex with degree outside of the desired range is at most $2n/n^{1000/27} \leqslant 1/n^{30}$, as desired. □

Proposition 5.4.5. *With probability at least $1 - 1/n^{30}$, G_1 has between $(1-\epsilon)p'm$ and $(1+\epsilon)p'm$ edges.*

Proof. For an edge $e \in E(G)$, let $X_e = 1$ if $e \in E(G_1)$ and $X_e = 0$ otherwise. Note that $\mathbb{P}[X_e = 1] = pq = p'$ for all e and that the X_es are independent. Let $X = \sum_{e \in E(G)} X_e$. Since G is Δ-regular, $\mathbb{E}[X] = p'm = p'(\Delta n/2)$. By Theorem 4.3.1,

$$\mathbb{P}[|X - p'm| \geqslant \epsilon p'm] \leqslant 2e^{-(p'\Delta n/2)\epsilon^2/3} = 2e^{-500n/\epsilon^2} < 1/n^{30}$$

as desired. $\square$

We are now ready to prove Lemma 5.4.3.

Proof of Theorem 5.4.3. For any $v \in V$, let d_v denote the degree of v in G_0. We say G_0 is *good* if and only if $d_v \in ((1-\epsilon/3)p\Delta, (1+\epsilon/3)p\Delta)$ for all $v \in V$. By Theorem 5.4.4, $\mathbb{P}[G_0 \text{ is good}] \geqslant 1 - 1/n^{30}$.

Let Y_v be a random variable denoting the degree of v in G_1. Since every edge in G_0 is included in G_1 independently with probability q, by Theorem 4.3.1, we have

$$\begin{aligned}\mathbb{P}[|Y_v - qd_v| \geqslant (\epsilon/3)qd_v \mid G_0 \text{ is good}] &\leqslant 2e^{-\epsilon^2 qd_v/27}\\ &\leqslant 2e^{-\epsilon^2(1-\epsilon/3)pq\Delta/27}\\ &\leqslant 2e^{-1500/(27\epsilon^2)}\end{aligned}$$

where we used $d_v \geqslant (1-\epsilon/3)p\Delta)$ since G_0 is good in the second inequality. Unfortunately we note that this probability is not low enough to allow us to union bound over all vertices in V. However, as G_0 has bounded maximum degree (when G_0 is good), the random variables $\{Y_v\}$ exhibit limited number of dependencies. We thus utilize concentration bounds for sums of dependent random variables to show that most vertices have the appropriate degree.

Let Z_v be an indicator variable denoting the event $|Y_v - qd_v| \geqslant (\epsilon/3)qd_v$. For convenience, let $\delta_0 := 2e^{-1500/(27\epsilon^2)}$. Then the previous inequality is equivalent to $\mathbb{P}[Z_v \mid G_0 \text{ is good}] \leqslant \delta_0$. Consider the collection of random variables $\mathcal{Z} = \{Z_v\}$. Given a graph G_0, variables Z_u and Z_v are dependent only when they are adjacent in G_0. Thus, if G_0 is good, then the dependency graph of $\mathcal{Z}$ has maximum degree at most $(1+\epsilon/3)p\Delta$; which implies that it has chromatic number at most $1 + (1+\epsilon/3)p\Delta \leqslant 4000 \log n/\epsilon^2$. Thus, applying Theorem 5.3.3 with $\lambda \leftarrow n\delta_0$, we have

$$\mathbb{P}\left[\sum_{v\in V} Z_v - \sum_{v \in V} \mathbb{E}[Z_v \,\middle|\, G_0 \text{ is good}] > n\delta_0 \mid G_0 \text{ is good}\right] \leqslant \exp\left(-\frac{2n\delta_0^2}{\chi(\mathcal{Z})}\right)$$

$$\mathbb{P}\left[\sum_{v\in V} Z_v > 2n\delta_0 \;\middle|\; G_0 \text{ is good}\right] \leqslant \exp\left(-\frac{2n\delta_0^2}{\chi(\mathcal{Z})}\right) \leqslant \exp\left(-\frac{2n\delta_0^2\epsilon^2}{4000\log n}\right)$$

Which is at most $1/n^{100}$. In particular, this implies that the probability that the number of vertices whose degree in G_1 is outside the range $\left((1\pm\epsilon/3)^2pq\Delta\right) \subset ((1\pm\epsilon)p'\Delta)$ exceeds $2n\delta_0$ is at most $\frac{1}{n^{100}}$ whenever G_0 is good. Let B denote the bad event that at least $2n\delta_0$ nodes in G_1 have degree outside the range $((1\pm\epsilon)p'\Delta)$. Then we have $\mathbb{P}[B \mid G_0 \text{ is good}] \leqslant 1/n^{100}$. Overall, without the conditioning, we have $\mathbb{P}[B] \leqslant \mathbb{P}[B \mid G_0 \text{ is good}] + \mathbb{P}[G_0 \text{ is not good}] \leqslant 1/n^{100} + 1/n^{30} \leqslant 2/n^{30}$. This completes the proof of the first statement in the lemma.

For the second part of the lemma, consider an arbitrary edge $e = \{u,v\} \in E(G_0)$ and let W_e be an indicator variable for the event that $e \in G_1$ *and* at least one of its end points u or v have their degree outside the range $((1-\epsilon)p'\Delta, (1+\epsilon)p'\Delta)$. Once again, let us condition on the event that G_0 is good. We note that $\mathbb{P}[W_e \mid G_0 \text{ is good}] \leqslant \mathbb{P}[e \in E(G_1) \text{ and } \max\{Z_u, Z_v\} = 1 \mid G_0 \text{ is good}]$. Further, the dependency graph of the collection of random variables $\mathcal{W} = \{W_e\}_{e\in E(G_0)}$ has maximum degree at most $(2(1+\epsilon/3)p\Delta)^2$ since W_e only depends on edges in the 2-neighborhood of edge e. Thus the chromatic number, $\chi(\mathcal{W}) \leqslant 1 + (2(1+\epsilon/3)p\Delta)^2 \leqslant 16(10^6)(\log^2 n)/\epsilon^4$. Once again, applying Theorem 5.3.3 with $\lambda \leftarrow \delta_0 q|E(G_0)|$, we have

$$\begin{aligned}&\mathbb{P}\left[\sum_{e\in E(G_0)} W_e > \delta_0 q|E(G_0)| + \sum_{e\in E(G_0)} \mathbb{E}[W_e|G_0 \text{ is good}] \;\middle|\; G_0 \text{ is good}\right]\\ \leqslant &\exp(-2|E(G_0)|q^2\delta_0^2\epsilon^4/(16(10)^6(\log^2 n))) \leqslant 1/n^{100}\end{aligned}$$

where we used $|E(G_0)| \geqslant n$ in the last inequality. Finally, we note that

$$\begin{aligned}&\mathbb{E}[W_e|G_0 \text{ is good}]\\ \leqslant&\mathbb{P}[\max(Z_u, Z_v) = 1|e \in E(G_1), G_0 \text{ is good}] \cdot \mathbb{P}[e \in E(G_1)|G_0 \text{ is good}]\\ \leqslant&2\mathbb{P}[Z_u = 1|e \in E(G_1), G_0 \text{ is good}] \cdot \mathbb{P}[e \in E(G_1)|G_0 \text{ is good}] \leqslant 2\delta_0 q\end{aligned}$$

Substituting into the inequality above, we get

$$\mathbb{P}\left[\sum_{e\in E(G_0)} W_e > 3\delta_0 q|E(G_0)| \;\middle|\; G_0 \text{ is good}\right] \leqslant 1/n^{100}$$

But, when G_0 is good, $|E(G_0| \leqslant (1+\epsilon/3)p\Delta n/2 < 2p\Delta n/2$, so

$$\mathbb{P}\left[\sum_{e\in E(G_0)} W_e > 6\delta_0 pq\Delta n/2 \;\middle|\; G_0 \text{ is good}\right] \leqslant 1/n^{100}$$

Overall, without the conditioning we have,

$$\begin{aligned}
&\mathbb{P}\left[\sum_{e\in E(G_0)} W_e > 6\delta_0 pq\Delta n/2\right]\\
\leqslant &\mathbb{P}\left[\sum_{e\in E(G_0)} W_e > 6\delta_0 pq\Delta n/2 \;\middle|\; G_0 \text{ is good}\right] + \mathbb{P}[G_0 \text{ is not good}]\\
\leqslant &\frac{1}{n^{100}} + \frac{1}{n^{30}} \leqslant \frac{2}{n^{30}}
\end{aligned}$$

The lemma now follows from a union bound over the two statements. □

Theorem 5.4.1 now follows directly from Theorem 5.4.2 and Theorem 5.4.3.

Proof of Theorem 5.4.1. If $\Delta \leqslant \frac{6000}{\epsilon^4}$, then we have $G_2 = G$ and the lemma follows trivially. Thus, we assume that $\Delta > \frac{6000}{\epsilon^4}$. By Theorem 5.4.5, $|E(G_1)| \geqslant (1-\epsilon)p'm$ with probability at least $1-1/n^{30}$. Thus, by the second statement of Theorem 5.4.3, the number of edges in G_1 that are not $(\epsilon, p'\Delta)$ balanced is at most $(12e^{-1500/27\epsilon^2})/(1-\epsilon)\cdot|E(G_1)| \leqslant 24e^{-1500/27\epsilon^2}|E(G_1)| < \epsilon|E(G_1)|$. At the same time, by the first statement of Theorem 5.4.3, at most $(4e^{-1500/(27\epsilon^2)})n \leqslant \epsilon|V(G_1)|$ vertices of G_1 have degree not in $((1-\epsilon)p'\Delta, (1+\epsilon)p'\Delta)$.

Substituting $\tau_v = \tau_e = \kappa = \epsilon$ in Theorem 5.4.2, we get that restricting to just the nodes with degrees in $((1-\epsilon)p'\Delta, (1+\epsilon)p'\Delta)$ must have a matching of size at least $(1-4\epsilon-\frac{1}{D+1})\frac{n}{2} \geqslant (1-5\epsilon)\frac{n}{2}$ where the last inequality used $\Delta > \frac{6000}{\epsilon^4}$. The probability bound follows from a union bound over the two lemmas. G_2 includes all these nodes because $\epsilon < 1/2$. □

5.4.2 Matching Stage

In this subsection, we give our matching stage algorithm which accepts a (low-degree) possibly-non-regular graph and returns an almost-perfect matching in a number of rounds which depends only on the maximum degree d and the desired accuracy $\epsilon > 0$, not on the number of nodes n. Combined with our sampling stage algorithm, which ensures that the maximum degree is a function of ϵ, we will be able to achieve an end-to-end number of rounds which depends only on ϵ and not the original degree Δ or the (original) number of nodes n. Our main result for this subsection is the following.

Lemma 5.4.6. *For $\epsilon > n^{-1/20}$, there is an $O(\epsilon^{-5} \log d)$-round randomized LOCAL algorithm where each vertex knows d and ϵ that returns a matching of size at least $(OPT - \epsilon n)$ on n-vertex graphs with maximum degree d with probability at least $1 - 1/n^{30}$. Furthermore, this algorithm is a CONGEST algorithm if ϵ and d are constant.*

To prove Theorem 5.4.6, we give an algorithm that improves a matching over many iterations; in each iteration, the algorithm attempts to find a large set of disjoint augmenting paths for the current matching. The algorithm finds these paths by constructing a hypergraph whose hyperedges each represent an augmenting path for the current matching. The algorithm finds an approximately maximum fractional matching in this hypergraph and then rounds that matching via independent random sampling and removing collisions. When the current matching is far from optimality, Theorem 5.3.2 certifies that this procedure actually finds a large matching. The algorithm is formally given by Algorithm 8, and it relies the following fractional matching algorithm for hypergraphs.

Theorem 5.4.7 (Theorem 4.8 of [70] with $\alpha = 2$ and using the $\delta(e)$s computed in the algorithm)**.** *In any f-bounded hypergraph $G = (V, E)$ with $\epsilon \in (0, 1)$ with maximum degree Δ, there is an $O(\log \Delta + f \log(f/\epsilon))$-round deterministic CONGEST algorithm for computing an $(f + \epsilon)$-approximate fractional hypergraph matching in G.*

Algorithm 8 ConstantMatch(G)

```
Input: An unweighted graph $G = (V, E)$; an accuracy parameter $\epsilon \in (n^{-1/20}, 1/2)$
Output: A matching $M$ in $G$
$M_0 \leftarrow \emptyset$
$k \leftarrow 4/\epsilon + 1$
$T \leftarrow 10^4/\epsilon^4 + 1$
$\tau \leftarrow 1/(4k^2)$
for $i \in \{1, 2, \ldots, T\}$ do
    $E_i \leftarrow$ the set of all $M_{i-1}$-augmenting paths $P$ with $|P| \leqslant k$
    $H_i \leftarrow (V, E_i)$, where each $P \in E_i$ yields a hyperedge between the vertices in $P$
    $x_i \leftarrow$ the fractional matching in $H_i$ given by Theorem 5.4.7 with $\epsilon = 1/2$
    /* $\mathcal{P}_i$ will contain disjoint $M_{i-1}$-augmenting paths we are choosing */
    $\mathcal{P}_i \leftarrow \emptyset$
    for each vertex $v \in V$ independently do
        $\mathcal{X}_v \leftarrow \{P \ \forall P \in E_i$ for which $v \in P\} \cup \{\star\}$
        $Y_v \leftarrow$ a randomly chosen member of $\mathcal{X}_v$, with $P \in \mathcal{X}_v$ chosen with probability $\tau x_i(P)$, and $\star$ chosen with probability $1 - \tau \sum_{Q \neq \star \in \mathcal{X}_v} x_i(Q)$
    end
    for each hyperedge $P \in E_i$ do
        Add $P$ to $\mathcal{P}_i$ if (a) there exists a $v \in P$ for which $Y_v = P$ and (b) for any $u \in V$ for which $Y_u \cap P \neq \emptyset$, $Y_u = P$
    end
    $M_i \leftarrow$ augmentation of $M_{i-1}$ by $\mathcal{P}_i$
end
return $M_T$
```

The remainder of this subsection is a proof of Theorem 5.4.6. We begin by verifying the following property of $\mathcal{P}_i$:

Proposition 5.4.8. *$\mathcal{P}_i$ is a collection of vertex-disjoint M_{i-1}-augmenting paths.*

Proof. By definition of E_i, $\mathcal{P}_i$ is a collection of M_{i-1}-augmenting paths, so it suffices to check that they are vertex-disjoint. Suppose, for the sake of contradiction, that there is a vertex $v \in V$ for which there exist $P_0, P_1 \in \mathcal{P}_i$ with $P_0 \neq P_1$ for which $v \in P_0$ and $v \in P_1$. By part (a) of the definition of $\mathcal{P}_i$, there exist $v_0 \in P_0$ and $v_1 \in P_1$ for which $Y_{v_0} = P_0$ and $Y_{v_1} = P_1$. By part (b) of the membership of P_0, since $P_0 \cap Y_{v_1} \neq \emptyset$, $Y_{v_1} = P_0$, a contradiction to the fact that $P_0 \neq P_1$. Thus, the sets in $\mathcal{P}_i$ are vertex-disjoint, as desired. $\square$

Note, first, that $|M_i| \geqslant |M_{i-1}|$, since all paths in $\mathcal{P}_i$ are augmenting paths. Let OPT denote the size of a maximum matching in G. If $|M_T| \geqslant OPT - \epsilon n$, then we are done, so assume for the sake of contradiction that $|M_T| < OPT - \epsilon n$. This means that $|M_i| \leqslant OPT - \epsilon n$ for all $i \in [T]$. We use this to show the following:

Proposition 5.4.9. *For any $i \in \{1, 2, \ldots, T\}$, if $|M_{i-1}| \leqslant OPT - \epsilon n$, then it holds that $\sum_{P \in E_i} x_i(P) \geqslant \frac{\epsilon n}{4(k+1)}$.*

Proof. By Proposition 5.3.2 applied to $M \leftarrow M_{i-1}$ and $\ell \leftarrow (k-1)/2$, there exists a collection $\mathcal{P}$ of M_{i-1}-augmenting paths with length at most k in G for which

$$\begin{aligned}|\mathcal{P}| &\geqslant \frac{1}{2}(OPT(1 - 2/(k-1)) - |M_{i-1}|) \\ &\geqslant \frac{1}{2}(OPT(1 - \epsilon/2) - (OPT - \epsilon n)) = \frac{\epsilon}{2}(n - OPT/2) \geqslant \frac{\epsilon n}{4}\end{aligned}$$

When the sets in $\mathcal{P}$ are viewed as hyperedges in H_i, $\mathcal{P}$ is a hypergraph matching thanks to the vertex disjointness of the sets. Thus, H_i has a hypergraph matching with at least $\frac{\epsilon n}{4}$ hyperedges. Since H_i is a k-bounded hypergraph, Theorem 5.4.7 implies that the total size of the fractional matching x_i is at least $|\mathcal{P}|/(k + 1/2) \geqslant \frac{\epsilon n}{4(k+1)}$ as desired. $\square$

We then use this lower bound to show that the rounding part of Algorithm 8 finds a large collection of augmenting paths. To analyze the sampling steps, we use McDiarmid's Inequality:

Theorem 5.4.10 ([220]). *Let $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n$ be sets, $c_1, c_2, \ldots, c_n \in \mathbb{R}$, and $f : \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n \to \mathbb{R}$ be a function with the property that, for any $i \in [n]$, $x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2, \ldots, x_n \in \mathcal{X}_n$, and $x_i' \in \mathcal{X}_i$,*

$$|f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n)| \leqslant c_i$$

Then, for any $\delta > 0$,

$$\mathbb{P}[|f(X_1, X_2, \ldots, X_n) - \mathbb{E}[f(X_1, X_2, \ldots, X_n)]| \geqslant \delta] \leqslant 2\exp\left(-\frac{2\delta^2}{\sum_{i=1}^n c_i^2}\right)$$

Fix an $i \in \{1, 2, \dots, T\}$ for the rest of this section. Define the function $f : \prod_{v \in V} \mathcal{X}_v \to \mathbb{R}$ to be $f(Y) := |\mathcal{P}_i|$, where Y is the n-tuple of Y_vs for all $v \in V$. This choice of function is inspired by Lemma 5.1 of [160]. We now prepare to use Theorem 5.4.10 by showing the following two results:

Proposition 5.4.11. *Let Y and Y' be two different tuples indexed by V for which there exists exactly one $v \in V$ for which $Y_v \neq Y'_v$. Then $|f(Y) - f(Y')| \leqslant 2k \leqslant 10/\epsilon$.*

Proof. Let $\mathcal{P}_i$ and $\mathcal{P}'_i$ be the sets resulting from Y and Y' respectively. It suffices to show that $||\mathcal{P}_i| - |\mathcal{P}'_i|| \leqslant k$ when $Y_v = \star$ and $Y'_v = P \neq \star$, as all remaining cases can be covered by the triangle inequality. Let $p_1, p_2, \dots, p_\ell$ with $\ell \leqslant k$ be the members of P. By part (b) of the definition, for all $j \in \{1, 2, \dots, \ell\}$, there exists at most one $P_j \in \mathcal{P}_i \cup \{\star\}$ for which $p_j \in P_j$. Every other set in $\mathcal{P}_i$ does not intersect P, so $\mathcal{P}_i \subseteq \mathcal{P}'_i \cup \{P_1, P_2, \dots, P_\ell\}$. Furthermore, $\mathcal{P}'_i \subseteq \mathcal{P}_i \cup \{P\}$. Therefore,

$$|\mathcal{P}_i| - \ell \leqslant |\mathcal{P}'_i| \leqslant |\mathcal{P}_i| + 1$$

as desired (since $\ell \leqslant k$). □

Proposition 5.4.12. *If $|M_{i-1}| \leqslant OPT - \epsilon n$, then*

$$\mathbb{E}_Y[f(Y)] \geqslant \frac{(1 - 2k^2\tau)\tau\epsilon n}{4(k+1)} \geqslant \frac{\epsilon^4 n}{5000}$$

Proof. We start by lower bounding the probability that any $P = \{p_1, p_2, \dots, p_\ell\} \in E_i$ is added to $\mathcal{P}_i$. First, note that

$$\begin{aligned}
&\mathbb{P}_Y[\text{there exists } j \in \{1, 2, \dots, \ell\} \text{ for which both } Y_{p_j} = P \text{ and } Y_{p_{j'}} = \star \text{ for all } j' \neq j] \\
&= \sum_{j=1}^{\ell} \tau x_i(P) \prod_{j' \neq j} \left(1 - \tau \sum_{Q \neq \star \in \mathcal{X}_{p_{j'}}} x_i(Q)\right) \\
&\geqslant \sum_{j=1}^{\ell} \tau x_i(P)(1 - \tau)^{\ell - 1} \\
&\geqslant \tau(1 - \tau)^k x_i(P)
\end{aligned}$$

since the Y_{p_j}s are independent. For any $u, w \in V$, let $E_i(u, w)$ denote the set of all hyperedges $Q \in E_i$ for which $u, w \in Q$. Note that for any $j \in \{1, 2, \dots, \ell\}$

$$\begin{aligned}
\mathbb{P}_Y[\text{for all } u \in V \setminus P, p_j \notin Y_u] &= \prod_{u \in V \setminus P} \mathbb{P}_{Y_u}[p_j \notin Y_u] \\
&= \prod_{u \in V \setminus P} \left(1 - \tau \sum_{Q \in E_i(u,p_j)} x_i(Q)\right) \\
&\geqslant 1 - \tau \sum_{u \in V} \sum_{Q \in E_i(u,p_j)} x_i(Q) \\
&= 1 - \tau \sum_{Q \in E_i : p_j \in Q} \sum_{u \neq p_j \in Q} x_i(Q) \\
&\geqslant 1 - k\tau \sum_{Q \in E_i : p_j \in Q} x_i(Q) \\
&\geqslant 1 - k\tau
\end{aligned}$$

By a union bound,

$$\mathbb{P}_Y[\text{for all } u \in V \setminus P, P \cap Y_u = \emptyset] \geqslant 1 - k^2\tau$$

The first event only depends on $V \setminus P$, while the second only depends on P. Let $\mathcal{E}*$ be the event in which there exists $j \in \{1, 2, \ldots, \ell\}$ for which both $Y_{p_j} = P$ and $Y_{p_{j'}} = \star$ and for all $u \in V \setminus P, P \cap Y_u = \emptyset$. Thus, by independence,

$$\mathbb{P}_Y[\mathcal{E}*] \geqslant (1 - k^2\tau)\tau(1-\tau)^k x_i(P)$$

Such P are added to $\mathcal{P}_i$, as $Y_{p_j} = P$ (satisfying condition (a)), $Y_u = \star$ for all $u \in P$ with $u \neq v_j$, and $Y_u \cap P = \emptyset$ for all $u \in V \setminus P$, so condition (b) is never triggered. Thus,

$$\mathbb{P}_Y[P \in \mathcal{P}_i] \geqslant (1 - k^2\tau)\tau(1-\tau)^k x_i(P) \geqslant (1 - 2k^2\tau)\tau x_i(P)$$

and

$$\begin{aligned}
\mathbb{E}_Y[f(Y)] &= \sum_{P \in E_i} \mathbb{P}_Y[P \in \mathcal{P}_i] \\
&\geqslant (1 - 2k^2\tau)\tau \sum_{P \in E_i} x_i(P)
\end{aligned}$$

$$\geqslant \frac{(1-2k^2\tau)\tau\epsilon n}{4(k+1)}$$

by Proposition 5.4.9, as desired. □

We are now ready to use McDiarmid's Inequality to lower bound the improvement in each iteration:

Proposition 5.4.13. $|\mathcal{P}_i| \geqslant \frac{\epsilon^4 n}{10000}$ *with probability at least* $1 - 2\exp\left(-\epsilon^6 n/10^{10}\right)$

Proof. By Theorem 5.4.10, Proposition 5.4.11, and Proposition 5.4.12, with $\delta \leftarrow \frac{\epsilon^4 n}{10000}$,

$$\mathbb{P}_Y[f(Y) \leqslant \frac{\epsilon^4 n}{10000}] \leqslant 2\exp\left(-\frac{2\delta^2}{(10/\epsilon)^2 n}\right) \leqslant 2\exp\left(-\epsilon^6 n/10^{10}\right)$$

as desired. □

Proof of Lemma 5.4.6. Let T' be the minimum value for which $|M_i| \geqslant OPT - \epsilon n$, or $T' = T+1$ if no such T' exists. T' is a random variable. We now upper bound the probability that $T' = T+1$. Since $|M_i| = |\mathcal{P}_i| + |M_{i-1}|$ for all i, $|M_T| \leqslant n$, and $|M_0| = 0$, there must exist an i for which $|\mathcal{P}_i| \leqslant \frac{n}{T} < \epsilon^4 n/10^4$. Thus,

$$\begin{aligned}
\mathbb{P}[T' = T+1] &= \mathbb{P}\left[T' = T+1 \text{ and } \exists i \in \{1,2,\ldots,T\} \text{ for which } |\mathcal{P}_i| < \frac{\epsilon^4 n}{10^4}\right] \\
&\leqslant \sum_{i=1}^{T} \mathbb{P}\left[T' = T+1 \text{ and } |\mathcal{P}_i| < \frac{\epsilon^4 n}{10^4}\right] \\
&\leqslant \sum_{i=1}^{T} \mathbb{P}\left[|M_{i-1}| < OPT - \epsilon n \text{ and } |\mathcal{P}_i| < \frac{\epsilon^4 n}{10^4}\right] \\
&\leqslant \sum_{i=1}^{T} \mathbb{P}\left[|\mathcal{P}_i| < \frac{\epsilon^4 n}{10^4} \;\middle|\; |M_{i-1}| < OPT - \epsilon n\right] \\
&\leqslant 2T\exp(-\epsilon^6 n/10^{10}) \\
&\leqslant \frac{1}{n^{30}}
\end{aligned}$$

where the second to last inequality follows from Proposition 5.4.13. Thus, with the desired probability, $T' < T+1$, in which case the algorithm finds a matching with the desired size.

The maximum degree of H_i is at most d^k, so each iteration of the for loop takes $\log(d^k) + k\log(k/\epsilon) = O((\log(d/\epsilon))/\epsilon)$, as all other operations takes a constant number of rounds in LOCAL, and a constant number of rounds in CONGEST when d and ϵ are constant. Thus, multiplying by T gives the desired runtime. □

5.4.3 The End-to-End Algorithm

In this subsection, we are finally ready to show that Theorem 5.1.1 follows from Theorem 5.4.1 and Theorem 5.4.6.

Proof of Theorem 5.1.1. We show that SamplingStage(G) (Algorithm 7) followed by Algorithm 8 returns the desired output in the desired runtime (for $\epsilon > n^{-1/20}$). By definition, the maximum degree of G_2 is at most $2p'\Delta = 6000/\epsilon^4$. We use this fact to bound both the runtime and the approximation error:

Runtime: Theorem 5.4.6 is applied to the G_2 produced by Algorithm 7, so $d = 6000/\epsilon^4$ in this case and $O(\epsilon^{-5}\log d) = O(\epsilon^{-5}\log(1/\epsilon))$ as desired. Since the runtime of Algorithm 7 is a constant number of rounds, the overall runtime is still just $O(\epsilon^{-5}\log(1/\epsilon))$. Note that the algorithm for Theorem 5.4.6 can be used, as each vertex in G_2 knows both d (which only depends on the original regular graph's degree Δ and ϵ) and ϵ.

Approximation: Let OPT denote the size of the maximum matching in the input graph G. By Theorem 5.4.1, we have that $OPT(G_2) \geqslant (1-5\epsilon)OPT$ with probability at least $1 - \frac{5}{n^{30}}$. We note that since G is Δ-regular, we have $OPT \geqslant (1 - \frac{1}{\Delta+1})\frac{n}{2} \geqslant \frac{n}{3}$.

Therefore, applying Lemma 5.4.6 results in a matching with size at least $(1 - 5\epsilon)OPT - \epsilon n \geqslant (1-5\epsilon)OPT - 3\epsilon OPT = (1-8\epsilon)OPT$ with probability at least $1 - 5/n^{30} - 1/n^{30} > 1 - 6/n^{30}$. The theorem now follows by replacing ϵ with $\epsilon/8$, which only changes run time by a constant factor. □

5.5 Exponentially Faster Algorithm in Dense Graphs

In this section, we show that there is an $O(\log(1/\epsilon))$-round algorithm to find a $(1+\epsilon)$-approximate maximum matching in Δ-regular graphs when $\Delta \geqslant (\frac{1}{\epsilon})^c$ for a large constant c. We note that the proof makes no attempt to optimize the constant c and we expect that it can be reduced significantly by a more careful analysis.

Theorem 5.1.2 (Main Result)**.** *Let $c = 10^5$ and let $0 < \epsilon < 1$ be a parameter. For Δ-regular graphs with $\Delta > (1/\epsilon)^c$, there is an $O(\log(1/\epsilon))$-round CONGEST algorithm that finds a $(1+\epsilon)$-approximate maximum matching with high probability.*

A Roadmap for the Proof of Theorem 5.1.2 The algorithm for proving Theorem 5.1.2 first runs a simple color coding step to find a bipartite almost regular subgraph, and then runs Luby's algorithm for $O(\log(1/\epsilon))$ rounds. Since our analysis for Luby's algorithm uses several martingale inequalities, it is cleaner to work with the sequential view of Luby that we present in Section 5.5.1. In Section 5.5.2 we show that a single round of Luby's algorithm in an almost regular graph matches a constant fraction of the nodes, with high probability.[8] In Section 5.5.3, we state our key recursive regularity lemma that shows that running one round of Luby's algorithm on an almost regular graph and deleting the matched nodes yields an almost regular graph. The *Local Recursive Regularity* lemma that bounds the probability that the degree of a particular node u almost exactly halves after each round is the most technical part of the proof and we devote Section 5.6 for its proof. Finally, in Section 5.5.4, we put these components together to finish the proof.

5.5.1 Sequential View of Luby's Algorithm

In the traditional distributed implementation of one round of Luby's algorithm, each edge e picks a uniformly random integer r_e in the set $\{1, 2, \ldots, M\}$ for an appropriately chosen polynomially large M. An edge e is chosen to be in the matching if $r_e < r_{e'}$ for all neighboring edges e'. This distributed view was given in Chapter 4 in the context of independent sets. We given it here again in the context of matchings for convenience (Algorithm 9).

[8] In fact, we prove a more general claim where it suffices that a constant fraction of the edges are balanced.

Algorithm 9 Luby

```
Input: An unweighted graph $G = (V, E)$, where $|E| = m$ and a constant $c' > 0$
Output: A matching $\mathcal{M}$
$I \leftarrow \emptyset$  for each edge $e \in E$ do
    $r_e \leftarrow$ uniformly random number in $\{1, 2, \ldots, 100m^{c'+2}\}$
end
for each edge $e \in E$ do
    Add $e$ to $\mathcal{M}$ if $r_e < r_{e'}$ for all neighboring edges $e'$ of $e$ in $G$ (i.e., for edges $e'$
     where $e \cap e' \neq \emptyset$)
end
return $\mathcal{M}$
```

Algorithm 10 SeqLuby

```
Input: an unweighted graph $G = (V, E)$
Output: a matching $\mathcal{M}$
$\mathcal{M} \leftarrow \emptyset$  $U \leftarrow E$  while $U \neq \emptyset$ do
    $e \leftarrow$ uniformly random element of the set $U$ $U \leftarrow U \setminus \{e\}$  if $\{e' \in E \mid e' \cap e \neq$
     $\emptyset\} \subseteq U$ then
        $\mathcal{M} \leftarrow \mathcal{M} \cup \{e\}$
    end
end
return $\mathcal{M}$
```

It is convenient for our analysis to work with a sequential view of Luby's algorithm. In the sequential view, the edges are sequentially sampled independently without replacement. When an edge e is sampled, it is added to the matching only if none of its neighboring edges had been sampled earlier. Crucially, unlike the greedy matching algorithm, a sampled edge e is *blocked* by a neighboring edge e' that was previously sampled even if e' itself is not in the matching. This sequential view was presented in Chapter 4, Section 4.5, and we present it here again in Algorithm 10 for convenience. In Chapter 4, Proposition 4.5.1, we showed that the two algorithms are equivalent by showing that they produce the same distribution over matchings with high probability[9].

[9]The Proposition was stated in the context of independent sets, but trivially extends to matchings.

Hence, in this chapter we focus on analyzing the sequential view of Luby whenever we want to make claims about a single round of Luby's algorithm. This incurs an additional tiny $1/\text{poly}(n)$ failure probability, which we can tolerate.

5.5.2 Analyzing One Round of Luby's Algorithm on Almost Regular Graphs

In this section we show that a single round of Luby's algorithm on almost regular graphs matches a constant fraction of the nodes with high probability. In fact, we prove this claim for a more general family of graphs in the following theorem.

Theorem 5.5.1. *Let G be an undirected graph with n nodes and m edges, and let $d = 2m/n$ be the average degree. Let $E^{low} = \{(u,v) \in E(G) \mid deg(u) \leqslant 2d, deg(v) \leqslant 2d\}$ be the set of edges induced by nodes with degree at most $2d$. If $|E^{low}| \geqslant m/2$, then one round of Luby's algorithm (Algorithm 9) finds a matching in G of size at least $n/288$ with high probability.*

Proof. For analysis, we'll focus on the sequential view of Luby's algorithm. By Proposition 4.5.1, the result holds for the distributed version as well.

Consider the first $t = n/24$ iterations of the while loop in Algorithm 10. We first claim that in each of these $t = n/24$ iterations, we add the sampled edge to the matching with probability at least $1/6$ (irrespective of previous random outcomes). Formally, for each $i \in [t]$, let $X_i \in \{0,1\}$ be a random variable indicating whether we add an edge to the matching in the ith iteration, and let $p_i = \mathbb{E}[X_i \mid X_1, \ldots X_{t-1}]$. We now show that $p_i \geqslant 1/6, \forall i \in [t]$.

We say that an edge e is *blocked* if during any of the first t iterations we sample some neighboring edge e'. By definition, if we sample some edge e in an iteration $i \in [t]$ and it isn't blocked, then e is always added to the matching. Next, we show that the number of blocked edges in E^{low} is at most $nd/6$. Assume for contradiction, that there is an iteration where edge $e = \{u,v\}$ is sampled that blocks more than $4d$ edges in E^{low}. This implies that either u or v is incident on more than $2d$ edges from E^{low}. But that's a contradiction since no edge from E^{low} can be incident on a node with degree $> 2d$. Thus, in each iteration, at most $4d$ edges from E^{low} can be blocked. Hence, in the first $t = n/24$ iterations, at most $(n/24) \cdot (4d) = nd/6$ edges are blocked in total.

Therefore, even at the end of the first t iterations, at least $|E^{low}| - nd/6 \geqslant nd/4 - nd/6 = nd/12$ edges from E^{low} remain unblocked. In any fixed iteration

$i \in [t]$, if we sample any of these edges, it will be added to the matching. Hence, the probability that we add an edge to the matching in iteration i is at least:

$$p_i \geqslant \frac{nd/12}{nd/2 - n/24} \geqslant \frac{nd/12}{nd/2} = 1/6$$

Finally, to prove the theorem we use the shifted martingale trick (that was discussed in the preliminaries, and is formally presented in Section 5.3.4) in a black-box fashion, as follows. Let $S_t = \sum_{i=1}^{t} X_i$, and let $p_i = \mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}]$. Since $p_i \geqslant 1/6$ for all $i \in [t]$, then by plugging in $P_i^\ell = 1/6$ for all i and $P^\ell = \sum_{i=1}^{t} P_i = t/6$ and $P^h = t$ in Theorem 5.3.12, we get that:

$$\mathbb{P}[S_t < n/288] \leqslant \exp\left(-\frac{(t/6 - n/288)^2}{16t}\right) \leqslant e^{-n/96}$$

□

5.5.3 Recursive Regularity

We first define some notation to facilitate the rest of the discussion. Intuitively, we say a node (α, Δ)-regular if all nodes in its two hop neighborhood have the same degree.

Definition 5.5.2 ((α, Δ)-regular node)**.** Let Δ be an integer and $\alpha \in [0, 1]$ be a real number. Given a graph G, we say that a node u is (α, Δ)-regular in G if for any $v \in \{u\} \cup N(u) \cup N^2(u)$, we have $\Delta(1-\alpha) \leqslant deg(v) \leqslant \Delta(1+\alpha)$.

We can now present our main technical lemma that states that if u is a (α, Δ)-regular node, then its degree becomes almost exactly $\Delta/2$ after running one round of Luby's algorithm, with failure probability exponentially small in Δ. The proof of Lemma 5.5.3 is the most technical part of this chapter and is deferred to Section 5.6.

Lemma 5.5.3 (Local Recursive Regularity Lemma)**.** *Let $\Delta \geqslant 2^{10}$ be an integer, and $\alpha \in [0, 1/10]$ be a real number. Let G be a bipartite graph and u be an (α, Δ)-regular node in it. Let $deg'(u)$ be the number of unmatched neighbors of u after running SeqLuby (Algorithm 10) on G. With probability at least $1 - \exp(-\Delta^{1/16})$, it holds that:*

$$\frac{\Delta}{2}\left(1 - (10\alpha + \Delta^{-1/600})\right) \leqslant deg'(u) \leqslant \frac{\Delta}{2}\left(1 + (10\alpha + \Delta^{-1/600})\right)$$

As long as Δ is large enough ($\approx \log n$), Theorem 5.5.3 followed by a union bound suffices to show that an almost regular graph remains almost regular with their degrees halved after one round of Luby's algorithm. However, for smaller Δ, we can no longer rely on a simple union bound. In the following lemma, we use the Chernoff-Hoeffding concentration inequality with bounded dependence to show that even for small Δ, *almost all* nodes halve their degree.

Lemma 5.5.4 (Recursive Regularity Lemma)**.** *Let $n \geqslant K \geqslant \Delta \geqslant 2^{10}$ be integers, and let $\alpha \in [0, 1/10]$, $\delta \in [0, 1/100]$ be real numbers. Let G be a bipartite graph with n nodes and max degree K and let G' be the graph obtained by running SeqLuby (Algorithm 10) on G and removing all matched nodes together with their incident edges.*

If at least $(1-\delta)$-fraction of nodes in G are (α, Δ)-regular, then at least $(1-\delta')$-fraction of nodes in G' are $(\alpha', \Delta/2)$-regular with probability at least

$$1 - \exp\left(-n/(K^{10}\exp(\Delta^{1/99}))\right)$$

where $\delta' = K^2 \cdot (\delta + 2\exp(-\Delta^{1/100})$ and $\alpha' = 10\alpha + \Delta^{-1/600}$.

Proof. We say that a node v is *good* if its degree in G' is in the range $\frac{\Delta}{2}(1 \pm \alpha')$. Any node that's not good is called *bad*. Let R be the set of nodes that are (α, Δ)-regular in G. By Lemma 5.5.3, each (α, Δ)-regular node in G that remains unmatched is good with probability at least $1 - \exp(-\Delta^{1/16})$. Hence, the expected number of nodes from R that are bad is at most $|R| \cdot \exp(-\Delta^{1/16}) \leqslant n$. To obtain a high probability bound, we use Chernoff-Hoeffding inequality with bounded dependence as follows.

For each node $u \in R$, let $B_u \in \{0, 1\}$ be an indicator random variable that indicates whether u is bad. We have $\sum_{u \in R} \mathbb{E}[X_u] \leqslant |R| \cdot \exp(-\Delta^{1/16}) \leqslant n \cdot \exp(-\Delta^{1/16})$. Since the maximum degree in G is K, each X_u depends on fewer than K^{10} nodes. Hence we apply Theorem 5.3.3 (with $\lambda = |R| \cdot \exp(-\Delta^{1/16})$) to get:

$$\begin{aligned}
\mathbb{P}\left[\sum_{u \in R} X_u > 2n\exp(-\Delta^{1/100})\right] &\leqslant \mathbb{P}\left[\sum_{u \in R} X_u > 2|R|\exp(-\Delta^{1/16})\right] \\
&\leqslant \mathbb{P}\left[\sum_{u \in R} X_u > \sum_{u \in R} \mathbf{E}[X_u] + |R|\exp(-\Delta^{1/16})\right] \\
&\leqslant \exp\left(-\frac{2(|R| \cdot \exp(-\Delta^{1/16}))^2}{|R| \cdot K^{10}}\right)
\end{aligned}$$

$$
\begin{aligned}
&= \exp\left(-\frac{2|R|}{K^{10}\exp\left(2\Delta^{1/16}\right)}\right)\\
&\leqslant \exp\left(-\frac{2|R|}{K^{10}\exp\left(\Delta^{1/99}\right)}\right)\\
&\leqslant \exp\left(-\frac{n}{K^{10}\exp\left(\Delta^{1/99}\right)}\right)
\end{aligned}
$$

where the last line used $\Delta > 2^{10}$ and $|R| \geqslant (1-\delta)n \geqslant n/2$. We say that a node v poisons a node u if v prevents u from being $(\alpha', \Delta/2)$-regular, i.e., v poisons u if v is *bad* and is at distance at most 2 from u. We note that each bad node poisons at most K^2 nodes. Let B be the set of bad nodes, i.e., we have $|B| = \sum_{u\in R} X_u$ and let $\tilde{B} = V(G) \setminus R$ be the set of nodes that are not (α, Δ) regular in G. For nodes in $\tilde{B}$, we have no guarantee on the probability of whether they become good, so we always assume that they poison K^2 nodes. In total, the number of nodes that are not $(\alpha', \Delta/2)$-regular in G' is at most $(|\tilde{B}| + |B|)K^2$ which is at most:

$$
\begin{aligned}
(|\tilde{B}| + 2n\cdot\exp\left(-\Delta^{1/100}\right))\cdot K^2 &\leqslant (\delta n + 2n\cdot\exp\left(-\Delta^{1/100}\right))\cdot K^2\\
&= n\cdot K^2\cdot(\delta + 2\exp\left(-\Delta^{1/100}\right))
\end{aligned}
$$

with probability at least $1 - \exp\left(-n/(K^{10}\exp\left(\Delta^{1/99}\right))\right)$, as desired. □

Theorem 5.5.4 shows that after each round of Luby's algorithm, the remaining graph still satisfies the requirement that most vertices remain almost regular, albeit with worsening parameters. The following technical claim shows that these parameters remain small enough after $O(\log(1/\epsilon))$ rounds.

Claim 5.5.5. Let Δ be an integer, $c = 1/10^5$, and ϵ be a real number satisfying $1 > \epsilon \geqslant 1/\Delta^c$. Furthermore let:

1. $\alpha_0 = \Delta^{-1/600}$ and $\alpha_i = 10\alpha_{i-1} + \Delta^{-1/600}$ for $i \geqslant 1$,
2. $\delta_0 = \exp(-\Delta^{1/200})$, and $\delta_i = \Delta^2(\delta_{i-1} + 2\exp(-(\Delta/2^i)^{1/100}))$ for $i \geqslant 1$.

It holds that $\alpha_i \leqslant 1/10$ and $\delta_i \leqslant \exp(-\Delta^{1/300})$ for $1 \leqslant i \leqslant 10\log(1/\epsilon)$.

Proof. We start with bounding α_i, observe that:

$$\alpha_i = 10\alpha_{i-1} + \Delta^{-1/600} \tag{1}$$

$$
\begin{aligned}
&= 10(10\alpha_{i-2} + \Delta^{-1/600}) + \Delta^{-1/600} && (2)\\
&\vdots && (3)\\
&\leqslant \sum_{j=0}^{i} 10^{j+1} \cdot \Delta^{-1/600} && (4)\\
&\leqslant i \cdot 10^{i+1} \cdot \Delta^{-1/600} && (5)\\
&\leqslant 100 \log(1/\epsilon) \cdot 10^{10\log(1/\epsilon)} \cdot \Delta^{-1/600} && (6)\\
&\leqslant \frac{100}{c} \log \Delta \cdot \Delta^{40/c} \cdot \Delta^{-1/600} && (7)\\
&\leqslant \Delta^{41/c-1/600} && (8)\\
&\leqslant \Delta^{41/98400-1/600} && (9)\\
&\leqslant \Delta^{1/2400-1/600} && (10)\\
&= \Delta^{-1/800} \leqslant 1/10 && (11)
\end{aligned}
$$

where (4) follows since $\alpha_0 = \Delta^{-1/600}$, (6) follows since $i \leqslant 10\log(1/\epsilon)$, and (7) follows since $\epsilon \geqslant 1/\Delta^{1/c}$. Next, we bound δ_i. Let $\Delta_i = \Delta/2^i$. Since $\epsilon \geqslant 1/(\Delta^{1/c})$ and $i \leqslant 10\log(1/\epsilon)$, we have that $\Delta_i = \Delta/2^i \geqslant \Delta/2^{10\log 1/\epsilon} \geqslant \Delta^{0.9}$. Therefore, $2\exp(-\Delta_i^{1/100}) \leqslant 2\exp(-\Delta^{9/1000}) \leqslant \exp(-\Delta^{1/200})$. Hence, we get that:

$$
\begin{aligned}
\delta_i &\leqslant \Delta^2(\delta_{i-1} + \exp(-\Delta^{1/200})) && (1)\\
&\leqslant \Delta^2(\Delta^2(\delta_{i-2} + \exp(-\Delta^{1/200})) + \exp(-\Delta^{1/200})) && (2)\\
&\vdots && (3)\\
&= \sum_{j=0}^{i} \Delta^{2(j+1)} \cdot \exp(-\Delta^{1/200}) && (4)\\
&\leqslant 2\exp(-\Delta^{1/200})\Delta^{2(i+1)} && (5)\\
&\leqslant 2\exp(-\Delta^{1/200}) \cdot \exp(\log^2 \Delta) && (6)\\
&\leqslant \exp(-\Delta^{1/300}) && (7)
\end{aligned}
$$

where (5) holds since $1/\Delta^2 \leqslant 1/2$, and (7) holds since $i \leqslant 10\log(1/\epsilon) \leqslant \log(\Delta)/10$.

□

We now extend Lemma 5.5.4 to a multi-round argument, showing that most of nodes' degrees after running Luby's algorithm for i rounds become $\approx \Delta/2^i$.

Lemma 5.5.6 (Multi-Round Recursive Regularity). *Let $(\Delta, \epsilon, (\alpha_i), (\delta_i))$ be as defined in Theorem 5.5.5. Let G be a bipartite n-node graph, where all but $\delta = \delta_0 = \exp\left(-\Delta^{1/200}\right)$-fraction of the nodes are (α_0, Δ)-regular. For $i \leqslant 10\log(1/\epsilon)$, let G_i be the graph obtained after running Luby's algorithm (Algorithm 9) for i rounds on G, where in each round we remove the matched nodes together with their incident edges. If $|V(G_i)| \geqslant \epsilon n$, then with high probability, at least $(1-\delta_i)$-fraction of the nodes in G_i are $(\alpha_i, \Delta/2^i)$-regular.*

Proof. The idea is to use a recursive argument where we apply Lemma 5.5.3 with a simple union bound in the dense case, and Lemma 5.5.4 in the sparse case. For convenience of notation, let $\Delta_i = \Delta/2^i$. Since we apply Lemmas 5.5.3 and 5.5.4 recursively for i rounds, we need to ensure that $\alpha_i \leqslant 1/10$ and $\delta_i \leqslant 1/100$, which follows from Claim 5.5.5. The proof is split into two cases, a dense case where $\Delta \geqslant \log^{50} n$, and a sparse case. We start with the dense case.

Dense case where $\Delta \geqslant \log^{50} n$: By Lemma 5.5.3 and a simple union bound argument, in the case where $\Delta \geqslant \log^{50} n$, *all* the nodes are $(\alpha_i, \Delta/2^i)$-regular after running Luby's algorithm (Algorithm 9) for i rounds with high probability. This is because the failure probability of Lemma 5.5.3 is at most $\exp\left(-\Delta_i^{1/16}\right)$ at round i, and $\Delta_i \geqslant \Delta^{0.99}$ for any i (since $\epsilon \geqslant 1/\Delta^{1/c}$ and $i \leqslant 10\log(1/\epsilon)$).

Sparse case where $\Delta \leqslant \log^{50} n$: The idea is to apply Lemma 5.5.4 recursively for i rounds and the proof follows by induction.

Induction base i=1: By Proposition 4.5.1 Algorithms 9 and 10 produce the same distributions over matchings in the graph G, up to $1/\mathrm{poly}(n)$ total variation distance. Hence, for $i = 1$ the claim follows directly from Lemma 5.5.4, since the failure probability of Lemma 5.5.4 is only $\exp\left(-n/\Delta^{10}\exp\left(\Delta^{1/99}\right)\right) < 1/n^{1000}$, for $\Delta \leqslant \log^{50} n$.

Induction step: Assume that the claim is true for i, i.e., assume that all but $|V(G_i)| \cdot \delta_i$ nodes in G_i are $(\alpha_i, \Delta/2^i)$-regular with high probability. We show that the claim is true for $i+1$. Again, by Proposition 4.5.1, Algorithms 9 and 10 produce the same distributions over matchings in the graph G_i, up to a $1/|E(G_i)|^c \leqslant 1/(|V(G_i)|)^c \leqslant 1/(\epsilon n)^c \leqslant (1/n^{0.99})^c$ total variation distance, for an arbitrarily large constant c. Since the max degree in G_i is at most Δ, by Lemma 5.5.4 it holds that all but $|V(G_{i+1})| \cdot \delta_{i+1}$ nodes in G_{i+1} are $(\alpha_{i+1}, \Delta/2^{i+1})$-regular in G_{i+1} with probability at least

$$1 - \exp\left(-\frac{|V(G_i)|}{\Delta^{10}\exp\left(\Delta_i^{1/99}\right)}\right) \geqslant 1 - \exp\left(-\frac{\epsilon n}{\Delta^{10}\exp\left(\Delta^{1/99}\right)}\right) \geqslant 1 - \frac{1}{n^{1000}}$$

where the first inequality holds since $\Delta_i \leqslant \Delta$ and $|V(G_i)| \geqslant \epsilon n$, and the second inequality holds since $\epsilon \geqslant 1/\Delta^{1/100} \geqslant 1/n^{1/100}$ and $\Delta \leqslant \log^{50} n$. Hence, by a union bound on all i's, we get that as long as $V(G_i)$ has at least ϵn nodes, it holds that all but $|V(G_i)| \cdot \delta_i$ nodes are (α_i, Δ_i)-regular in G_i with probability at least $1 - 1/n^{100}$. □

5.5.4 Putting it together

Theorem 5.5.6 shows that the graph remains almost regular after $i \approx \log(1/\epsilon)$ repeated applications of Luby's algorithm, and Theorem 5.5.1 showed that each round of Luby's on an almost regular graph matches a constant fraction of nodes. We can now combine these two results to show that as long as the remaining graph is large enough, each application of Luby's algorithm matches a constant fraction of nodes.

We first present a corollary of Theorem 5.5.1 to formalize that the almost regular graphs implied by Theorem 5.5.6 do indeed satisfy the requirements of Theorem 5.5.1.

Corollary 5.5.7. *Let $\Delta \geqslant C$, where C is a large enough constant, and let G' be a graph with n' nodes and max degree Δ. Suppose at least $(1 - \exp(-\Delta^{1/300}))$-fraction of the nodes are (α', Δ')-regular for $\alpha' \leqslant 1/10$. It holds that Luby's algorithm matches $n'/288$ nodes in G' with high probability.*

Proof. To apply Theorem 5.5.1, we need to show that at least $1/2$ of the edges in G' are incident with nodes of degree smaller than $2\bar{d}$, where $\bar{d}$ is the average degree in G'. We start with bounding $\bar{d}$. Let R be the set of nodes that are (α', Δ')-regular in G'.

Upper bounding $\bar{d}$: Observe that:

$$\begin{aligned}\bar{d} &= \frac{\sum_{v \in V(G')} deg_{G'}(v)}{|V(G')|} \\ &\leqslant \frac{|R| \cdot \Delta'(1+\alpha') + \exp(-\Delta^{1/300})|V(G')| \cdot \Delta}{n'} \\ &\leqslant \Delta'(1+\alpha') + \exp(-\Delta^{1/300}) \cdot \Delta \\ &\leqslant \Delta'(1+2/10)\end{aligned}$$

Lower bounding $\bar{d}$: Observe that:

$$\begin{aligned}\bar{d} &= \frac{\sum_{v \in V(G')} deg(v)}{|V(G')|} \geqslant \frac{|R| \cdot \Delta'(1-\alpha')}{|V(G')|} \\ &\geqslant (1-\exp(-\Delta^{1/300}))\Delta'(1-\alpha') \geqslant (1-1/10)\Delta'(1-1/10) \\ &\geqslant \Delta'(1-2/10)\end{aligned}$$

where we used that $\exp(-\Delta^{1/300}) < 1/10$ for Δ larger than C (which is a sufficiently large constant).

Hence, all the nodes that are (α', Δ')-regular in G' have degree within a factor of 2 from the average degree. Furthermore, the only edges that are incident to nodes with degree larger than $2\bar{d}$ are the edges incident to the nodes that are not (α', Δ')-regular, and these are only a $\Delta \cdot \exp\left(-\Delta^{1/300}\right)) << 1/2$-fraction of the edges. The claim follows. □

We are now ready to prove Theorem 5.1.2.

***Proof of Theorem* 5.1.2.** We first provide a proof that assumes that the graph in bipartite, and then we lift this assumption by a simple sampling argument. We simply run Luby's algorithm in G for $i = 10\log(1/\epsilon)$ rounds. By Lemma 5.5.6, all but an $\exp\left(-\Delta^{1/300}\right)$-fraction of the nodes in the graph are (α_i, Δ_i)-regular at round i with high probability. Hence, by Theorem 5.5.7, in each of these rounds we match at least a $(1/288)$-fraction of the nodes. Therefore, after $O(\log(1/\epsilon))$, all but ϵ-fraction of the nodes are matched, as desired.

Overcoming bipartiteness: Lemma 5.5.6 assumes that the input graph is bipartite. To overcome this, we apply a simple color-coding trick. Before running Luby's algorithm, each node picks a uniformly random color independently in $\{0,1\}$. This naturally defines a bipartite subgraph graph G' by ignoring all mono-colored edges. Observe that for a given node $u \in G$, the degree of u in G' is $\Delta(1 \pm \Delta^{-0.4})$ with probability at least $1 - \exp\left(-\Delta^{0.2}/6\right)$ by a standard Chernoff-Hoeffding bound (Theorem 4.3.1). Hence, if $\Delta \geqslant \log^{50} n$, then all the nodes degrees in G' are $\Delta(1 \pm \Delta^{-0.4})$ with high probability. Therefore, all the nodes are also $(\Delta^{-0.4}, \Delta)$-regular in that case. Otherwise, if $\Delta < \log^{50} n$, then we can use a bounded dependence Chernoff-Hoeffding (Theorem 5.3.3) type argument, as follows. By a union bound, the probability that a node u isn't $(\Delta^{-0.4}, \Delta)$-regular is at most $\Delta^2 \cdot \exp\left(-\Delta^{0.2}/6\right) \leqslant \exp\left(-\Delta^{0.1}\right)$. Hence, the expected number of nodes that aren't $(\Delta^{-0.4}, \Delta)$-regular is at most $n \cdot \exp\left(-\Delta^{0.1}\right)$. Furthermore, the event of whether a node is $(\Delta^{-0.4}, \Delta)$-regular depends only on $< \Delta^{10}$ other nodes. Hence, by applying Theorem 5.3.3 with $\lambda = n \cdot \Delta^{-0.1}$, we get that with high probability, all but $n \cdot \exp\left(-\Delta^{1/200}\right)$ nodes are $(\Delta^{-0.4}, \Delta)$-regular in G'. Hence, since $\Delta^{-0.4} \leqslant \Delta^{-1/600}$, it suffices to apply Lemma 5.5.6 on G'. This concludes the proof. □

5.6 Local Recursive Regularity

We devote this section to the proof of Theorem 5.5.3 that we restate here for convenience.

Lemma 5.6.1 (Local Recursive Regularity Lemma). *Let $\Delta \geqslant 2^{10}$ be an integer, and $\alpha \in [0, 1/10]$ be a real number. Let G be a bipartite graph and u be an (α, Δ)-regular node in it. Let $deg'(u)$ be the number of unmatched neighbors of u after running SeqLuby (Algorithm 10) on G. With probability at least $1 - \exp(-\Delta^{1/16})$, it holds that:*

$$\frac{\Delta}{2}\left(1 - (10\alpha + \Delta^{-1/600})\right) \leqslant deg'(u) \leqslant \frac{\Delta}{2}\left(1 + (10\alpha + \Delta^{-1/600})\right)$$

5.6.1 Notation and Preliminaries

For the entirety of this section, let u be some fixed (α, Δ)-regular node in G. Recall that $N(u)$ is the set of neighbors of u and $N^d(u)$ is the set of nodes at distance

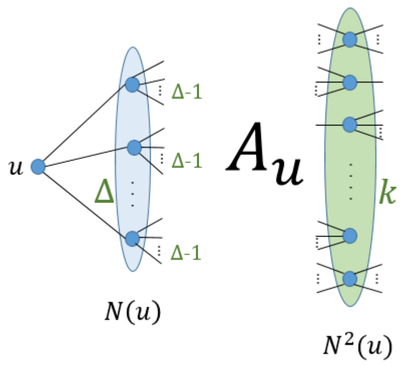

Figure 5.1: We focus on a node u that has Δ neighbors. The set of neighbors of u is denoted by $N(u)$, and the set of nodes at distance 2 from u is denoted by $N^2(u)$. We denote the size of $N^2(u)$ by k. The set of edges between $N(u)$ and $N^2(u)$ is denoted by A. Observe that each $v \in N(u)$ has exactly $\Delta - 1$ edges incident to it in A_u. However, this is not necessarily the case for the nodes in $N^2(u)$, that can have anywhere between 1 to Δ incident edges in A_u.

exactly d from u. Let $k = |N^2(u)|$. Let E_u denote the set of edges that are at most 3 hops away from u, i.e,

$$E_u = E \cap \Big\{(\{u\} \times N(u)) \cup (N(u) \times N^2(u)) \cup (N^2(u) \times N^3(u))\Big\}$$

and let $A_u = E \cap \{N(u) \times N^2(u)\}$. The lemma states that almost exactly $\Delta/2$ edges from A_u are matched with high probability.

It is easy to observe that only edges from E_u can affect the insertion of edges in A_u in the matching, and hence in this section, we restrict our attention to only edges in the set E_u. In other words, instead of SeqLuby (Algorithm 10), it suffices to show the lemma for Algorithm 11. We formalize this in the following observation.

Observation 5.6.2. Let D_0 and D_1 be the distributions over the matching edges picked from $A_u = N(u) \times N^2(u)$ by Algorithms 10 and 11, respectively. It holds that for any matching $\mathcal{M} \subseteq A_u$:

$$\mathbb{P}_{\mathcal{M}' \sim D_0}[\mathcal{M}' = \mathcal{M}] = \mathbb{P}_{\mathcal{M}' \sim D_1}[\mathcal{M}' = \mathcal{M}]$$

Throughout this section, by iteration i we mean the ith iteration of the while loop in Algorithm 11. We say an edge $e \in A_u$ *survives* in iteration i if neither it nor any of its neighbors have been sampled in the first $i-1$ iterations. Intuitively, a surviving edge that gets sampled gets added to the resulting matching.

Algorithm 11 SeqLuby$_u$ - SeqLuby in the neighborhood of u

```
Input: an unweighted graph $G = (V, E)$ and a node $u \in V$
Output: a matching $\mathcal{M}_u \subseteq N(u) \times N^2(u)$
$\mathcal{M}'_u \leftarrow \emptyset$  $E_u \leftarrow \{(u \times N(u)) \cup (N(u) \times N^2(u)) \cup (N^2(u) \times N^3(u))\} \cap E$  $U \leftarrow E_u$
 while $U \neq \emptyset$ do
    $e \leftarrow$ uniformly random element of the set $U$   $U \leftarrow U \setminus \{e\}$  if $\{e' \mid e' \cap e \neq \emptyset\} \subseteq U$ then
        $\mathcal{M}'_u \leftarrow \mathcal{M}'_u \cup \{e\}$
    end
end
$\mathcal{M}_u = \mathcal{M}'_u \cap (N(u) \times N^2(u))$  return $\mathcal{M}_u$
```

Definition 5.6.3 (The set of surviving edges E_i)**.** For $i \geqslant 1$, we say that an edge $e \in A_u$ is still surviving at the beginning of the ith iteration of the while loop in Algorithm 11, if it holds that

$$\{e\} \cup \{e' \mid e' \cap e \neq \emptyset\} \subseteq U$$

where U is the set of edges that haven't been sampled so far. Let E_i be the set of surviving edges at the beginning of the ith iteration.

The Stopping Time: Since our proofs involve martingales analysis, one important aspect of it is *the stopping time t* of our martingales. In several applications of martingale inequalities, we need to stop the martingale earlier than its final termination point. This concept is standard in martingales analysis, and in most cases, without stopping the martingale early, one wouldn't be able to apply the black-box martingale inequalities conveniently. In our context, t is the number of iterations of the while loop in Algorithm 11 that we consider for our analysis. We set $t = k \log \Delta / 100$ (recall that $k = |N^2(u)|$). For the lower bound on $|\mathcal{M}_u|$, we show that the size of $|\mathcal{M}_u|$ is already large enough after t iterations of the while loop. For the upper bound, we show that $|\mathcal{M}_u|$ doesn't change substantially after the first t iterations (this is formally proven in Lemma 5.6.11), which allows us to "stop" the analysis after t iterations.

For easy reference, we include a table of notations that we use throughout the proof in Table 5.1.

Table 5.1: Table of Notations

Notation	Meaning
V	The set of nodes in the input graph.
E	The set of edges in the input graph.
u	The fixed node throughout Section 5.6.
$N(u)$	The set of neighbors of u.
$N^d(u)$	The set of nodes at distance d from u.
k	The size of $N^2(u)$, i.e., $k = \|N^2(u)\|$.
Δ	The degree of the original input regular graph. In the Local Recursive Regularity Lemma, all the nodes in $\{u\} \cup N(u) \cup N^2(u)$ have degree in the range $\Delta(1 \pm \alpha)$.
α	A parameter in $[0, 1/200]$ used to lower and upper bound the degrees of the nodes in $\{u\} \cup N(u) \cup N^2(u)$ by $\Delta(1 \pm \alpha)$.
E_u	The set of edges in $(u \times N(u)) \cup (N(u) \times N^2(u)) \cup (N^2(u) \times N^3(u))$.
A_u	The set of edges in $N(u) \times N^2(u)$.
$\mathcal{M}_u$	The returned set of matching edges in A_u, i.e., $\|\mathcal{M}_u\|$ is the number of matched neighbors of u.
U	The set of edges that weren't sampled so far in Algorithm 11.
Z_i	A boolean random variable indicating whether we add an edge from A_u to the matching in the ith iteration.
q_i	The probability that $Z_i = 1$ given $Z_1, \cdots, Z_{i-1}$. Since the Z_i's are boolean, $q_i = \mathbb{E}[Z_i \mid Z_1, \cdots, Z_{i-1}]$.
E_i	The set of surviving edges in A_u at the beginning of the ith iteration.
t	The stopping time $t = k \log \Delta / 100$.

5.6.2 Analyzing Number of Surviving Edges $|E_i|$

Our first goal is to obtain tight bounds on the number of surviving edges at the beginning of the ith iteration. Towards this goal, we first bound the total number of edges adjacent to any node that are sampled by the first t iterations. We setup some additional notation to facilitate this discussion.

Definition 5.6.4 (Labeled Edges)**.** We say that an edge e is labeled at iteration $i \in [t]$ if e has been sampled in one of the first i iterations, i.e., if $e \notin U$ at the end of ith iteration of the while loop in Algorithm 11. If e is labeled at iteration i then it also labeled at any iteration $j > i$.

Claim 5.6.5. With probability at least $1 - \exp\left(-\Delta^{1/11}\right)$, every node in $\{u\} \cup N(u) \cup N^2(u)$ has at most $\Delta^{1/9}$ labeled edges incident to it at any iteration $i \in [t = k \log \Delta / 100]$.

Proof. Fix a node $v \in \{u\} \cup N(u) \cup N^2(u)$. We would like to prove the claim by applying Theorem 5.3.11. For this, let $X_i \in \{0, 1\}$ be a random variable indicating whether we sample an edge incident to v at iteration i, and let $p_i(v) = \mathbb{E}[X_i \mid X_1, \cdots, X_{i-1}] = \mathbb{P}[X_i = 1 \mid X_1, \cdots, X_{i-1}]$. Furthermore, let $S_j = \sum_{i=1}^{j} X_i$ be the number of labeled edges incident to v at iteration j. Since $S_j \geqslant S_i$ for any $j > i$, it suffices to prove the claim for $j = t$. Observe that since the degree of any node in $\{u\} \cup N(u) \cup N^2(u)$ is at least $(1-\alpha)\Delta$, it holds that $|E_u| \geqslant (1-\alpha)\Delta(k+1)$. Hence, we have that:

$$p_i(v) \leqslant \frac{(1+\alpha)\Delta}{(1-\alpha)\Delta(k+1) - (i-1)}$$

This is because v has at most $(1+\alpha)\Delta$ edges incident to it, and at the i'th iteration we're sampling an edge out of $|E_u| - (i-1) \geqslant (1-\alpha)\Delta(k+1) - (i-1)$ remaining edges. Therefore, we have that:

$$\begin{aligned} p_i(v) &\leqslant \frac{(1+\alpha)\Delta}{(1-\alpha)\Delta(k+1) - (i-1)} \\ &\leqslant \frac{(1+\alpha)\Delta}{(1-\alpha)\Delta(k+1) - t + 1} \\ &= \frac{(1+\alpha)\Delta}{(1-\alpha)\Delta(k+1) - k\log\Delta/100 + 1} \\ &\leqslant \frac{8}{k+1} \end{aligned}$$

Let $P_i = 8/(k+1)$, and let $P = \sum_{i=1}^{t} P_i = 8t/(k+1) \leqslant \log \Delta$. By plugging these values into Theorem 5.3.11 with $\lambda = \Delta^{1/9}$ and $M = 1$, we get that:

$$\mathbb{P}[S_t \geqslant \Delta^{1/9}] \leqslant \exp\left(-\frac{(\Delta^{1/9} - \log\Delta)^2}{8\log\Delta + 2(\Delta^{1/9} - \log\Delta)/3}\right) \leqslant \exp\left(-\Delta^{1/10}\right)$$

as desired.

The claim now follows by a union bound over all vertices in $\{u\} \cup N(u) \cup N^2(u)$. □

We say that node a v is labeled at iteration i if the edge sampled during the ith iteration is adjacent to v. We now utilize Theorem 5.6.5 to argue that any iteration, every node v is equally likely to be labeled with probability $\approx 1/k$.

Lemma 5.6.6. *Let $p_i(v)$ be the probability that we label a node v in the ith iteration. With probability at least $1 - \exp\left(-\Delta^{1/11}\right)$, for any node $v \in \{u\} \cup N(u) \cup N^2(u)$ and for any iteration $i \in [t = k\log\Delta/100]$, it holds that the probability that v is labeled at iteration i is:*

$$\frac{(1-\alpha)}{(1+\alpha)\cdot k} \cdot (1 - \Delta^{-7/8}) \leqslant p_i(v) \leqslant \frac{(1+\alpha)}{(1-\alpha)\cdot k} \cdot (1 + \frac{\log\Delta}{25\Delta})$$

Proof. We start with the upper bound. The probability that we label an edge incident to v at any iteration i is at most:

$$\begin{aligned} p_i(v) &\leqslant \frac{(1+\alpha)\Delta}{(1-\alpha)\Delta(k+1) - (i-1)} \leqslant \frac{(1+\alpha)\Delta}{(1-\alpha)\Delta(k+1) - t + 1} \\ &\leqslant \frac{(1+\alpha)\Delta}{(1-\alpha)\Delta k - k\log\Delta/100} = \frac{(1+\alpha)\Delta}{(1-\alpha)\Delta k(1 - \log\Delta/(100(1-\alpha)\Delta))} \\ &\leqslant \frac{(1+\alpha)}{(1-\alpha)\cdot k} \cdot (1 + \frac{\log\Delta}{25\Delta}) \end{aligned}$$

where the last inequality follows since:

$$\begin{aligned} \frac{1}{1 - \log\Delta/(100(1-\alpha)\Delta)} &= \frac{1 - \log\Delta/(100(1-\alpha)\Delta) + \log\Delta/(100(1-\alpha)\Delta)}{1 - \log\Delta/(100(1-\alpha)\Delta)} \\ &\leqslant 1 + \frac{\log\Delta/(100(1-\alpha)\Delta)}{1/2} \leqslant 1 + \frac{\log\Delta}{25\Delta} \end{aligned}$$

For the lower bound, we apply Corollary 5.6.5, which says that with probability at least $1 - \exp(-\Delta/11)$, every node $v \in \{u\} \cup N(u) \cup N^2(u)$ has at most $\Delta^{1/9}$ labeled edges incident to it at any of the first t iterations. Hence, any node v has at least $(1-\alpha)\Delta - \Delta^{1/9}$ unlabeled edges incident to it at any iteration $i \in [t]$. The probability that we label v at iteration i is exactly the probability that we pick one of these unlabeled edges out of the $|E_u| - (i-1)$ remaining edges. Hence:

$$\begin{aligned} p_i(v) &\geqslant \frac{(1-\alpha)\Delta - \Delta^{1/9}}{|E_u| - (i-1)} && (1)\\ &\geqslant \frac{(1-\alpha)\Delta - \Delta^{1/9}}{(1+\alpha)\Delta(k+1) - (i-1)} && (2)\\ &\geqslant \frac{(1-\alpha)\Delta \cdot (1 - 1/((1-\alpha)\Delta^{8/9}))}{(1+\alpha)\Delta(k+1)} && (3)\\ &\geqslant \frac{(1-\alpha)\Delta \cdot (1 - 2/\Delta^{8/9})}{(1+\alpha)\Delta(k+1)} && (4)\\ &\geqslant \frac{(1-\alpha)}{(1+\alpha)\cdot k} \cdot (1 - \Delta^{-7/8}) && (5) \end{aligned}$$

where (1) holds since the degree of v is at least $(1-\alpha)\Delta - \Delta^{1/9}$, (2) holds since $|E_u| \leqslant (1+\alpha)\Delta(k+1)$, (4) holds since $\alpha \leqslant 1/10$, which implies that $(1 - 1/((1-\alpha)\Delta^{8/9})) \geqslant (1 - 2\Delta^{-8/9})$, and (5) holds since $1/(k+1) \geqslant (1/k)(1-1/k)$, and since $k \geqslant \Delta/7$. The latter holds since u has at least $(1-\alpha)\Delta$ neighbors, and each of them has at least $(1-\alpha)\Delta - 1$ edges incident to it in $|A_u|$. Hence:

$$k = |N^2(u)| \geqslant \frac{|A_u|}{(1+\alpha)\Delta} \geqslant \frac{(1-\alpha)^2\Delta^2 - (1-\alpha)\Delta}{(1+\alpha)\Delta} \geqslant \Delta/7$$

where the last inequality holds since $\alpha \leqslant 1/10$. □

The above bounds on $p_i(v)$ allow us to show that the expected number of surviving edges drops by a constant factor in each iteration. Recall that an edge $e \in A_u$ is said to be surviving at iteration i if neither it or any of its neighbors have been sampled before that iteration. Informally, we show that $\mathbb{E}[|E_i| \mid E_{i-1}] \approx (1 - 2/k)|E_{i-1}|$.

Lemma 5.6.7. *With probability at least* $1 - \exp(-\Delta^{1/11})$, *for any* $1 < i \leqslant t = k \log \Delta/100$, *it holds that:*

1. $\mathbb{E}[|E_i| \mid E_1, \cdots, E_{i-1}] \geqslant \left(1 - \frac{2}{k}\frac{(1+\alpha)}{(1-\alpha)}\left(1 + \Delta^{-6/7}\right)\right) \cdot |E_{i-1}|$.

2. $\mathbb{E}[|E_i| \mid E_1, \cdots, E_{i-1}] \leqslant \left(1 - \frac{2}{k}\frac{(1-\alpha)}{(1+\alpha)}\left(1 - \Delta^{-6/7}\right)\right) \cdot |E_{i-1}|$.

Proof. We say that an edge e' is killed at the $(i-1)$'th iteration if $e' \in E_{i-1}$, and either e' or a neighboring edge is sampled in the $(i-1)$th iteration. That is, $i-1$ is the first iteration in which e' is no longer surviving. We denote the number of killed edges at the $(i-1)$'th iteration by K_{i-1}. Observe that $|E_i| = |E_{i-1}| - K_{i-1}$. We analyze $\mathbb{E}[K_{i-1} \mid E_{i-1}]$ to derive $\mathbb{E}[|E_i| \mid E_{i-1}]$.

Some notation: For an edge $e = \{v, w\}$, let $p_{i-1}(e)$ be the probability that e is sampled at the $(i-1)$th iteration. Similarly, for a set of edges S, let $p_{i-1}(S)$ be the probability that some edge in S is sampled in the $(i-1)$th iteration. We slightly abuse notation and we also denote by $p_{i-1}(v)$ the probability that a node v is labeled in the $(i-1)$th iteration. That is, $p_{i-1}(v)$ is the probability that the sampled edge in the $(i-1)$th iteration is incident to v. Since the graph is bipartite, we assume without loss of generality that $v \in L$ and $w \in R$ when we refer to an edge $\{v, w\}$, where L and R are the left and right sides of the graph, respectively. Furthermore, we assume without loss of generality that $N(u) \subseteq L$ and $N^2(u) \subseteq R$. Observe that when an edge e is sampled, it kills the set of edges $\{e' \in E_{i-1} \mid e \cap e' \neq \emptyset\}$. Let $Y_e \in \{0, 1\}$ be 1 if and only if $e \in E_{i-1}$, and let $d_{i-1}(v) \subseteq E_{i-1}$ be the number of surviving edges incident to a node v at the $(i-1)$th iteration. Observe that:

$$\mathbb{E}[K_{i-1} \mid E_{i-1}] = \sum_{e=\{v,w\}} p_{i-1}(e) \cdot |\{e' \in E_{i-1} \mid e \cap e' \neq \emptyset\}| \tag{1}$$

$$= \sum_{e=\{v,w\}} p_{i-1}(e) \cdot \left(d_{i-1}(v) + d_{i-1}(w) - Y_e\right) \tag{2}$$

$$= \left(\sum_{v \in L} p_{i-1}(v) \cdot d_{i-1}(v)\right) + \left(\sum_{w \in R} p_{i-1}(w) \cdot d_{i-1}(w)\right) - p_{i-1}(E_{i-1}) \tag{3}$$

$$= \left(\sum_{v \in N(u)} p_{i-1}(v) \cdot d_{i-1}(v)\right) + \left(\sum_{w \in N^2(u)} p_{i-1}(w) \cdot d_{i-1}(w)\right) - p_{i-1}(E_{i-1}) \tag{4}$$

where (2) follows by a simple inclusion/exclusion argument, since the number of edges that are killed by $e = \{v, w\}$ is the number of edges killed that are incident to v, plus the number of edges killed that are incident to w, minus the number of edges killed that are incident to both (which could only be e itself if happens to be surviving at the beginning of the $(i-1)$th iteration), and (4) follows since the only nodes in L that are incident to edges in E_{i-1} are the nodes in $N(u)$ and the only nodes in R that are incident to edges in E_{i-1} are the nodes in $N^2(u)$. Now we are ready to derive an upper and lower bounds on $\mathbb{E}[K_{i-1} \mid E_{i-1}]$.

Upper Bound on $\mathbb{E}[K_{i-1} \mid E_{i-1}]$: By plugging the upper bound on $p_{i-1}(v)$ and $p_{i-1}(w)$ from Lemma 5.6.6 into equation (4) above, we get that:

$$\mathbb{E}[K_{i-1} \mid E_{i-1}] \leqslant \frac{2(1+\alpha)}{(1-\alpha)\cdot k} \cdot |E_{i-1}| \cdot (1 + \frac{\log \Delta}{25\Delta}) \leqslant \frac{2(1+\alpha)}{(1-\alpha)\cdot k} \cdot |E_{i-1}| \cdot (1+\Delta^{-6/7})$$

Lower Bound on $\mathbb{E}[K_{i-1} \mid E_{i-1}]$: First, observe that since $i \leqslant t = k \log \Delta / 100$, we have that:

$$\begin{aligned} p_{i-1}(E_{i-1}) &\leqslant \frac{|E_{i-1}|}{(1-\alpha)\Delta(k+1) - (i-2)} \\ &\leqslant \frac{|E_{i-1}|}{(1-\alpha)(\Delta k) - t} \\ &\leqslant \frac{|E_{i-1}|}{(1-\alpha)(\Delta k)/2} \\ &\leqslant \frac{4|E_{i-1}|}{\Delta k} \end{aligned}$$

where the last inequality holds since $\alpha \geqslant 1/2$. Hence, by plugging the lower bound on $p_{i-1}(v)$ and $p_{i-1}(w)$ from Lemma 5.6.6 into equation (4) above, we get that:

$$\begin{aligned} \mathbb{E}[K_{i-1} \mid E_{i-1}] &\geqslant \frac{2(1-\alpha)}{(1+\alpha)\cdot k} \cdot |E_{i-1}| \cdot (1-\Delta^{-7/8}) - \frac{4|E_{i-1}|}{\Delta k} \\ &\geqslant \frac{2(1-\alpha)}{(1+\alpha)\cdot k} \cdot |E_{i-1}| \cdot (1-\Delta^{-6/7}) \end{aligned}$$

Deriving Upper and Lower Bounds on $\mathbb{E}[|E_i| \mid E_{i-1}]$: Since $|E_i| = E_{i-1} - K_{i-1}$, we get that:

$$\mathbb{E}[|E_i| \mid E_{i-1}] \geqslant \left(1 - \frac{2(1+\alpha)}{(1-\alpha)\cdot k}\left(1 + \Delta^{-6/7}\right)\right) \cdot |E_{i-1}|$$

and that:

$$\mathbb{E}[|E_i| \mid E_{i-1}] \leqslant \left(1 - \frac{2(1-\alpha)}{(1+\alpha)\cdot k}\left(1 - \Delta^{-6/7}\right)\right) \cdot |E_{i-1}|$$

as desired. Finally, the lemma follows since $\mathbb{E}[|E_i| \mid E_1, \cdots, E_{i-1}] = \mathbb{E}[|E_i| \mid E_{i-1}]$. □

Now we are ready to give concentration results for $|E_i|$ for any $i \in [t = k\log\Delta/100]$. We start with showing a lower for $|E_i|$ in Lemma 5.6.8 and then we show an upper bound on $|E_i|$ in Lemma 5.6.9.

Lemma 5.6.8. *Let $\gamma_\ell = \left(1 - \frac{2}{k}\frac{(1+\alpha)}{(1-\alpha)}\left(1+\Delta^{-6/7}\right)\right)$. With probability at least $1 - \exp\left(-\Delta^{1/13}\right)$, it holds that for every $i \in [t = k\log\Delta/100]$.*

$$|E_i| \geqslant \gamma_\ell^{i-1} \cdot ((1-\alpha)\Delta)^2 \cdot (1 - \Delta^{-1/5})$$

Proof. The idea of the proof is to use the scaled martingale trick that was briefly discussed in Section 5.2.2. Let $\mathcal{B}$ be the event in which the upper and lower bounds on $\mathbb{E}[|E_i| \mid E_{i-1}]$ from Lemma 5.6.7 hold. By Lemma 5.6.7, the event $\mathcal{B}$ happens with probability at least $1 - \exp\left(-\Delta^{1/11}\right)$. Our analysis next is conditioned on the event $\mathcal{B}$ happening. For $i \geqslant 1$, we define the following random variable F_i as follows.

$$F_i = \frac{|E_i|}{\gamma_\ell^{i-1}}$$

Observe that F_i is a supermartingale (see also Definition 5.3.6) with a starting point $F_1 = |E_1| = |A_u|$ (recall that $A_u = (N(u) \times N^2(u)) \cap E$ is the set of edges between u's neighbors and their neighbors). This is because:

$$\mathbb{E}[F_i \mid F_1, \cdots, F_{i-1}] = \frac{1}{\gamma_\ell^{i-1}} \mathbb{E}[|E_i| \mid E_1, \cdots, E_{i-1}] \geqslant \frac{|E_{i-1}|}{\gamma_\ell^{i-2}} = F_{i-1}$$

where the first equality follows from Observation 5.3.4, and the inequality follows from Theorem 5.6.7. Hence, to get a lower bound on $|E_i|$, we would like to apply the supermartingale inequality from Theorem 5.3.9. For this, we analyze $Var[F_i \mid F_{i-1}]$ and $\mathbb{E}[F_i \mid F_{i-1}] - F_i$ for $i \geqslant 2$. First, observe that for any $i \in [t]$, we have that:

$$\gamma_\ell^i \geqslant \gamma_\ell^t = \left(1 - \frac{2}{k}\frac{(1+\alpha)}{(1-\alpha)}\left(1 + \Delta^{-6/7}\right)\right)^t \geqslant \left(1 - \frac{8}{k}\right)^{k \log \Delta / 100} \geqslant \Delta^{-1/10}$$

where the second to last inequality follows from the fact that $1 - x \geqslant e^{-2x}$ for any $0 \leqslant x \leqslant 1/2$. Next, for $i \geqslant 2$, we analyze $Var[F_i \mid F_{i-1}]$ and then $\mathbb{E}[F_i \mid F_{i-1}] - F_i$. Observe that:

$$\begin{aligned}
Var[F_i \mid F_{i-1}] &= Var\left[Fi - \frac{F_{i-1}}{\gamma_\ell} \mid F_{i-1}\right] && (1)\\
&= Var\left[\frac{|E_i|}{\gamma_\ell^{i-1}} - \frac{|E_{i-1}|}{\gamma_\ell^{i-1}} \mid E_{i-1}\right] && (2)\\
&\leqslant \frac{1}{\gamma_\ell^{i-1}} \mathbb{E}\left[(|E_i| - |E_{i-1}|)^2 \mid E_{i-1}\right] && (3)\\
&\leqslant \frac{2\Delta}{\gamma_\ell^{i-1}} \mathbb{E}\left[||E_i| - |E_{i-1}|| \mid E_{i-1}\right] && (4)\\
&= \frac{2\Delta}{\gamma_\ell^{i-1}} \left(\mathbb{E}[|E_{i-1}| \mid E_{i-1}] - \mathbb{E}[|E_i| \mid E_{i-1}]\right) && (5)\\
&\leqslant \frac{2\Delta}{\gamma_\ell^{i-1}} \left(|E_{i-1}| - |E_{i-1}| + 12|E_{i-1}|/k\right) && (6)\\
&\leqslant \frac{2\Delta}{\gamma_\ell^{i-1}} \cdot \frac{12|E_{i-1}|}{k} && (7)\\
&\leqslant \frac{96\Delta^{3.1}}{k} && (8)
\end{aligned}$$

where (1) follows from Observation 5.3.5, (3) follows by the definition of variance, (4) follows since the maximum value of $|E_i - E_{i-1}|$ is $(1+\alpha)\Delta \leqslant 2\Delta$, (5) follows since $|E_{i-1}| \geqslant |E_i|$, (6) follows since $\mathbb{E}[|E_i| \mid E_{i-1}] \geqslant (1-12/k)|E_{i-1}|$ by Lemma 5.6.7, and (8) follows since $\gamma_\ell^{i-1} \geqslant \Delta^{-1/10}$ and $|E_{i-1}| \leqslant |E_1| \leqslant (1+\alpha)^2\Delta^2 \leqslant 4\Delta^2$.

Next, we analyze $\mathbb{E}[F_i \mid F_{i-1}] - F_i$ for any $i \geqslant 2$. Observe that:

$$\begin{aligned}
\mathbb{E}[F_i \mid F_{i-1}] - F_i &= \frac{1}{\gamma_\ell^{i-1}}(\mathbb{E}[|E_i| \mid E_{i-1}] - |E_i|) && (1)\\
&\leqslant \frac{1}{\gamma_\ell^{i-1}}(|E_{i-1}| - |E_i|) && (2)\\
&\leqslant 4\Delta^{1.1} && (3)
\end{aligned}$$

where (2) holds since $|E_i| \leqslant |E_{i-1}|$ (as there are more surviving edges at iteration $i-1$ compared to iteration i), and (3) holds since $|E_{i-1}| - |E_i| \leqslant 2(1+\alpha)\Delta \leqslant 4\Delta$ (as we kill at most $2(1+\alpha)\Delta - 1$ edges in each iteration), and $\gamma_\ell^{i-1} \geqslant \Delta^{-1/10}$. Furthermore, observe that $F_1 = |E_1| = |A_u| \geqslant ((1-\alpha)\Delta)^2 - (1-\alpha)\Delta$, as u has at least $(1-\alpha)\Delta$ neighbors, and each of them has at least $(1-\alpha)\Delta - 1$ edges incident to it in A_u. Hence, we can plug our bounds on $Var[F_i \mid F_{i-1}]$ and $\mathbb{E}[F_i \mid F_{i-1}] - F_i$ into Theorem 5.3.9 to get that for a given $i \in [t]$, conditioned on the event $\mathcal{B}$ happening, we have that:

$$\begin{aligned}
\mathbb{P}[F_i \leqslant ((1-\alpha)\Delta)^2 - (1-\alpha)\Delta - \Delta^{7/4} \mid \mathcal{B}] &\leqslant \mathbb{P}[F_i \leqslant F_1 - \Delta^{7/4} \mid \mathcal{B}) && (1)\\
&\leqslant \exp\left(-\frac{\Delta^{3.5}}{2\Delta^{3.1}\log\Delta + 8\Delta^{2.85}}\right) && (2)\\
&\leqslant \exp\left(-\Delta^{3/10}\right) && (3)
\end{aligned}$$

where (2) follows since $i \leqslant t = k\log\Delta/100$, and therefore $\sum_{j=1}^{i} 96\Delta^{3.1}/k \leqslant t \cdot 96\Delta^{3.1}/k \leqslant \Delta^{3.1}\log\Delta$. Furthermore, observe that $\mathbb{P}[F_i \leqslant ((1-\alpha)\Delta)^2(1-\Delta^{-1/5})] \leqslant \mathbb{P}[F_i \leqslant ((1-\alpha)\Delta)^2 - (1-\alpha)\Delta - \Delta^{7/4}]$. This is because $((1-\alpha)\Delta)^2(1-\Delta^{-1/5}) \leqslant ((1-\alpha)\Delta)^2 - (1-\alpha)\Delta - \Delta^{7/4}$. Hence, to get rid of the conditioning on the event $\mathcal{B}$ happening, we just use a union bound argument to get that for a given $i \in [t]$:

$$\begin{aligned}
\mathbb{P}[F_i \leqslant ((1-\alpha)\Delta)^2 \cdot (1-\Delta^{-1/5})] &\leqslant \mathbb{P}[F_i \leqslant ((1-\alpha)\Delta)^2 - (1-\alpha)\Delta - \Delta^{7/4} \mid \mathcal{B}]\\
&\leqslant \exp\left(-\Delta^{3/10}\right) + \exp\left(-\Delta^{1/11}\right)
\end{aligned}$$

$$\leqslant \exp\left(-\Delta^{1/12}\right)$$

By another simple union bound argument, we get that with probability at least $1 - \exp(-\Delta^{1/13})$, *for every* $i \in [t]$, $F_i \geqslant ((1-\alpha)\Delta)^2 \cdot (1 - \Delta^{-1/5})$. Hence, the lemma follows since $F_i = |E_i|/\gamma_\ell^{i-1}$ □

Lemma 5.6.9. *Let* $\gamma_h = \left(1 - \frac{2}{k}\frac{(1-\alpha)}{(1+\alpha)}\left(1 - \Delta^{-6/7}\right)\right)$. *With probability at least* $1 - \exp(-\Delta^{1/13})$, *it holds that for every* $i \in [t = k\log\Delta/100]$.

$$|E_i| \leqslant \gamma_h^{i-1} \cdot ((1+\alpha)\Delta)^2(1+\Delta^{-1/5})$$

Proof. The proof is very similar to the proof of Lemma 5.6.8. As in the proof on Lemma 5.6.8, we define the random variable:

$$F_i = \frac{|E_i|}{\gamma_h^{i-1}}$$

Observe that F_i is a submartingale (see also Definition 5.3.6) with starting point $F_1 = |E_1| = |A_u|$. This is because for any $i \geqslant 2$, we have that:

$$\mathbb{E}[F_i \mid F_1, \cdots, F_{i-1}] = \frac{1}{\gamma_h^{i-1}}\mathbb{E}[|E_i| \mid E_1, \cdots, E_{i-1}] \leqslant \frac{|E_{i-1}|}{\gamma_h^{i-2}} = F_{i-1}$$

Hence, we would like to use Theorem 5.3.10 to get a concentration result for F_i. For this, we need to analyze $Var[F_i \mid F_{i-1}]$ and $F_i - \mathbb{E}[F_i \mid F_{i-1}]$. Observe that by a similar calculation to the one that we provided for γ_ℓ^i in the proof of Lemma 5.6.8, it holds that $\gamma_h^i \geqslant \Delta^{-1/10}$, for any $i \in [t]$. Hence, by the exact same calculation of $Var[F_i \mid F_{i-1}]$ in the proof of Lemma 5.6.8, it holds that $Var[F_i \mid F_{i-1}] \leqslant 96\Delta^{3.1}/k$. It remains to analyze $F_i - \mathbb{E}[F_i \mid F_{i-1}]$. Observe that:

$$\begin{aligned}
F_i - \mathbb{E}[F_i \mid F_{i-1}] &= \frac{1}{\gamma_h^{i-1}}(|E_i| - \mathbb{E}[|E_i| \mid E_{i-1}]) && (1)\\
&\leqslant \frac{1}{\gamma_h^{i-1}}(|E_{i-1}| - |E_{i-1}| + 12|E_{i-1}|/k) && (2)\\
&\leqslant \frac{48\Delta^2}{k\gamma_h^{i-1}} && (3)\\
&\leqslant 336\Delta^{1.1} && (4)
\end{aligned}$$

where (2) follows since $|E_i| \leqslant |E_{i-1}|$ and $\mathbb{E}[|E_i| \mid E_{i-1}] \geqslant 1 - 12|E_{i-1}|/k$, where the latter follows from Lemma 5.6.7, (3) follows since $|E_{i-1}| \leqslant |E_1| \leqslant ((1+\alpha)\Delta)^2 \leqslant 4\Delta^2$, and (4) follows since $k \geqslant \Delta/7$ and $\gamma_h^{i-1} \geqslant \Delta^{-0.1}$.

The rest of the proof is exactly as the proof of Lemma 5.6.8, where instead of using Theorem 5.3.9, we use Theorem 5.3.10 to get concentration for F_i, and then deduce the desired bound for $|E_i|$. Observe that our analysis above is conditioned on the event $\mathcal{B}$ happening, where $\mathcal{B}$ is the event in which the bounds on $\mathbb{E}[|E_i| \mid E_{i-1}]$ hold. By Theorem 5.3.10, we have that:

$$\begin{aligned}\mathbb{P}[F_i \geqslant ((1+\alpha)\Delta)^2 + \Delta^{7/4} \mid \mathcal{B}] &\leqslant \mathbb{P}[F_i \geqslant F_1 + \Delta^{7/4} \mid \mathcal{B}] \\ &\leqslant \exp\left(-\Delta^{3/10}\right)\end{aligned}$$

Finally, by two simple applications of a union bound argument, the probability that there exists an $i \in [t]$ for which $F_i \geqslant ((1+\alpha)\Delta)^2 + \Delta^{7/4}$ is at most $\exp(-\Delta^{1/13})$.

Hence, with probability at least $1 - \exp(-\Delta^{-1/3})$ for every $i \in [t]$, $F_i \leqslant ((1+\alpha)\Delta)^2 + \Delta^{7/4} \leqslant ((1+\alpha)\Delta)^2(1+\Delta^{-1/5})$. The proof of the lemma follows since $F_i = |E_i|/\gamma_h^{i-1}$

□

5.6.3 A Lower Bound on $|\mathcal{M}_u|$

Recall that by definition, whenever a surviving edge gets sampled at some iteration i, it gets added to the matching $\mathcal{M}_u$. We can now utilize the sharp concentration bounds on the number of surviving edges to obtain concentration bounds on the size of the matching $\mathcal{M}_u$. We first give a lower bound on $|\mathcal{M}_u|$ in this section and give an upper bound in the next.

Lemma 5.6.10. *With probability at least* $1 - \exp(-\Delta^{-1/15})$ *it holds that:*

$$|\mathcal{M}_u| \geqslant \frac{\Delta}{2} \cdot \frac{(1-\alpha)^3}{(1+\alpha)^2} \cdot (1 - \Delta^{-1/300})$$

Proof. It suffices to show that the size of $\mathcal{M}_u$ is large enough after $t = k \log \Delta/100$ iterations of the while loop in Algorithm 11, as the size of $\mathcal{M}_u$ can only increase after the tth iteration. Let $\mathcal{B}$ be the event in which the upper and lower bounds on $|E_1|, \cdots, |E_t|$ from Lemma 5.6.8 and Lemma 5.6.9 hold. Observe that the event $\mathcal{B}$ happens with probability at least $1 - \exp(-\Delta^{1/14})$. We prove the lemma conditioned on the event $\mathcal{B}$ happening and then get rid of this condition by a simple union bound argument. To prove the lemma in the case where the event $\mathcal{B}$ happens we apply Theorem 5.3.12, as follows. Recall that Z_i is an indicator random variable indicating whether we add an edge to $\mathcal{M}_u$ in the ith iteration of the while loop in Algorithm 11, and recall that q_i is a random variable representing the probability that we add an edge in the ith iteration given the randomness so far. In other words: $q_i = \mathbb{P}[Z_i = 1 \mid Z_1, \cdots, Z_{i-1}] = \mathbb{E}[Z_i \mid Z_1, \cdots, Z_{i-1}]$. As discussed in Section 5.2.2, we have that:

$$q_i = \frac{|E_i|}{|E_u| - (i-1)}$$

This is because the probability that we add an edge to the matching in the ith iteration is the probability that we pick a surviving edge among the remaining $|E_u| - (i-1)$ edges. Next, we show an upper and a lower bound on $\sum_{i=1}^{t} q_i$, conditioned on the event $\mathcal{B}$ happening. Observe that $|E_u| \leqslant ((1+\alpha)\Delta)(k+1)$, as the number of edges in E_u is exactly the number of edges touching the nodes in $N^2(u)$ plus the number of edges touching u itself. Since each node has degree at most $(1+\alpha)\Delta$, we get at most $(1+\alpha)\Delta(k+1)$ edges in total in E_u (recall that $|N^2(u)| = k$).

A lower bound on $\sum_{i=1}^{t} q_i$. Let $\gamma_\ell = \left(1 - \frac{2}{k}\frac{(1+\alpha)}{(1-\alpha)}\left(1 + \Delta^{-6/7}\right)\right)$. Observe that:

$$\begin{aligned}
\sum_{i=1}^{t} q_i &\geqslant \sum_{i=1}^{t} \frac{|E_i|}{(1+\alpha)\Delta(k+1) - (i-1)} && (1)\\
&\geqslant \sum_{i=1}^{t} \frac{\gamma_\ell^{i-1} \cdot ((1-\alpha)\Delta)^2 \cdot (1 - \Delta^{-1/5})}{(1+\alpha)\Delta(k+1)} && (2)\\
&= \frac{((1-\alpha)\Delta)^2 \cdot (1 - \Delta^{-1/5})}{(1+\alpha)\Delta(k+1)} \sum_{i=1}^{t} \gamma_\ell^{i-1} && (3)\\
&= \frac{(1-\alpha)^2\Delta \cdot (1 - \Delta^{-1/5})}{(1+\alpha)(k+1)} \cdot \frac{1 - \gamma_\ell^t}{1 - \gamma_\ell} && (4)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{(1-\alpha)^2\Delta\cdot(1-\Delta^{-1/5})}{(1+\alpha)(k+1)}\cdot\frac{1-\left(1-\frac{2}{k}\frac{(1+\alpha)}{(1-\alpha)}\left(1+\Delta^{-6/7}\right)\right)^t}{1-\left(1-\frac{2}{k}\frac{(1+\alpha)}{(1-\alpha)}\left(1+\Delta^{-6/7}\right)\right)} && (5)\\
&\geqslant \frac{(1-\alpha)^2\Delta\cdot(1-\Delta^{-1/5})}{(1+\alpha)(k+1)}\cdot\frac{1-\Delta^{-1/100}}{\frac{2}{k}\frac{(1+\alpha)}{(1-\alpha)}\left(1+\Delta^{-6/7}\right)} && (6)\\
&\geqslant \frac{(1-\alpha)^3\Delta\cdot k}{2(1+\alpha)^2(k+1)}\cdot\frac{(1-\Delta^{-1/5})(1-\Delta^{-1/100})}{(1+\Delta^{-6/7})} && (7)\\
&\geqslant \frac{\Delta}{2}\cdot\frac{(1-\alpha)^3}{(1+\alpha)^2}\cdot(1-\Delta^{-1/200}) && (8)
\end{aligned}
$$

where (2) follows from Lemma 5.6.8, (4) follows from the geometric series formula, (6) follows since $\gamma_\ell^t \leqslant \Delta^{-1/100}$, and (8) follows since $k \geqslant \Delta/7$ and from a simple arithmetic. Now we show an upper bound on $\sum_{i=1}^t q_i$.

An upper bound on $\sum_{i=1}^t q_i$. Let $\gamma_h = \left(1-\frac{2}{k}\frac{(1-\alpha)}{(1+\alpha)}\left(1-\Delta^{-6/7}\right)\right)$. Observe that $|E_u| \geqslant ((1-\alpha)\Delta)k$, by just counting the degrees of the nodes in $N^2(u)$. Hence, we have that:

$$
\begin{aligned}
\sum_{i=1}^t q_i &= \sum_{i=1}^t \frac{|E_i|}{|E_u|-(i-1)} && (1)\\
&\leqslant \sum_{i=1}^t \frac{\gamma_h^{i-1}\cdot((1+\alpha)\Delta)^2(1+\Delta^{-1/5})}{(1-\alpha)\Delta k - t} && (2)\\
&= \frac{((1+\alpha)\Delta)^2(1+\Delta^{-1/5})}{(1-\alpha)\Delta k - k\log\Delta/100}\sum_{i=1}^t \gamma_h^{i-1} && (3)\\
&\leqslant \frac{((1+\alpha)\Delta)^2(1+\Delta^{-1/5})(1+\Delta^{-8/10})}{(1-\alpha)\Delta k}\sum_{i=1}^t \gamma_h^{i-1} && (4)\\
&= \frac{(1+\alpha)^2\Delta(1+\Delta^{-1/5})(1+\Delta^{-8/10})}{(1-\alpha)k}\frac{1-\gamma_h^t}{1-\gamma_h} && (5)\\
&\leqslant \frac{(1+\alpha)^2\Delta(1+\Delta^{-1/5})(1+\Delta^{-8/10})}{(1-\alpha)k}\cdot\frac{1}{1-\left(1-\frac{2}{k}\frac{(1-\alpha)}{(1+\alpha)}\left(1-\Delta^{-6/7}\right)\right)} && (6)
\end{aligned}
$$

$$\leqslant \frac{(1+\alpha)^2\Delta(1+\Delta^{-1/5})(1+\Delta^{-8/10})}{(1-\alpha)k} \cdot \frac{1}{\frac{2}{k}\frac{(1-\alpha)}{(1+\alpha)}\left(1-\Delta^{-6/7}\right)} \tag{7}$$

$$\leqslant \frac{\Delta}{2} \cdot \frac{(1+\alpha)^3}{(1-\alpha)^2} \cdot \frac{(1+\Delta^{-1/3})(1+\Delta^{-8/10})}{(1-\Delta^{-6/7})} \tag{8}$$

$$\leqslant \frac{\Delta}{2} \cdot \frac{(1+\alpha)^3}{(1-\alpha)^2} \cdot (1+\Delta^{-1/100}) \tag{9}$$

where (2) follows from Lemma 5.6.9 and the rest follows from simple arithmetic. Having proved the upper and lower bounds on $\sum_{i=1}^t q_i$, we apply Theorem 5.3.12 with $X_i = Z_i$, $S_t = \sum_{i=1}^t Z_i \leqslant |\mathcal{M}_u|$, $p_i = q_i$, $P^\ell = \frac{\Delta}{2} \cdot \frac{(1-\alpha)^3}{(1+\alpha)^2} \cdot (1-\Delta^{-1/200})$, $P^h = \frac{\Delta}{2} \cdot \frac{(1+\alpha)^3}{(1-\alpha)^2} \cdot (1+\Delta^{-1/100})$, $M = 1$, and $\lambda = P^\ell - \Delta^{0.6}$ to get that:

$$\mathbb{P}(|M_u| \leqslant \lambda \mid \mathcal{B}) \leqslant \exp\left(-\frac{\Delta^{1.2}}{100\Delta}\right) \leqslant \exp\left(-\Delta^{1/10}\right)$$

Finally, since the event $\mathcal{B}$ happens with probability at least $1 - \exp -\Delta^{1/14}$, by a simple union bound, we get that $|\mathcal{M}_u| \geqslant P^\ell - \Delta^{0.6} \geqslant \frac{\Delta}{2} \cdot \frac{(1-\alpha)^3}{(1+\alpha)^2} \cdot (1-\Delta^{-1/300})$ with probability at least $1 - \exp\left(-\Delta^{1/15}\right)$, as desired. □

5.6.4 An Upper Bound on $|\mathcal{M}_u|$

Observe that all our analysis takes into consideration only the first $t = k \log \Delta/100$ iterations of the while loop in Algorithm 11. To show an upper bound on $|\mathcal{M}_u|$, it doesn't suffice to consider only the first t iterations as we also need to make sure that not too many edges are added to $|\mathcal{M}_u|$ after the tth iteration. Our proof for the upper bound on $|\mathcal{M}_u|$ is split into two parts. First, in Lemma 5.6.11, we show that Algorithm 11 doesn't add too many more edges to $|\mathcal{M}_u|$ after the tth iteration. Then, in Lemma 5.6.12, we show an upper bound on the number of edges that are added to $|\mathcal{M}_u|$ up to the tth iteration, and deduce the upper bound on $|\mathcal{M}_u|$. Recall that we say that a node is labeled at iteration i if one of its incident edges was sampled in one of the first i iterations. Observe that a labeled node at iteration i can't be matched at iteration $j > i$, as all of its incident edges are non-surviving at iteration j (see also Definition 5.6.3). In the following lemma, we show that the number of unlabeled neighbors of u after the t's iteration is small.

Lemma 5.6.11. *With probability at least $1 - \exp(-\Delta^{1/12})$, the number of unlabeled nodes in $N(u)$ after the tth iteration is processed is at most $\Delta^{0.9975}$.*

Proof. We use the scaled martingale. Let $\mathcal{B}$ be the event in which the upper and lower bounds on the probability that a node is labeled from Lemma 5.6.6 hold. We first provide an analysis conditioned on the event $\mathcal{B}$ happening, and then we remove this assumption by a simple union bound argument. By Lemma 5.6.6, the event $\mathcal{B}$ happens with probability at least $1 - \exp(-\Delta^{1/11})$. Let W_i be the number of unlabeled nodes in $N(u)$ after the ith iteration is processed. Observe that by Lemma 5.6.6, we have that:

$$\mathbb{E}[W_i \mid W_{i-1}] \leqslant \left(1 - \frac{1-\alpha}{(1+\alpha)k} \cdot (1 - \Delta^{-7/8})\right) W_{i-1}$$

This is because by Lemma 5.6.6, each node v is labeled with probability at least $\frac{1-\alpha}{(1+\alpha)k} \cdot (1 - \Delta^{-7/8})$ at any iteration. Therefore, the total probability mass that the unlabeled nodes have at the beginning of the ith iteration is $W_{i-1} \cdot \frac{1-\alpha}{(1+\alpha)k} \cdot (1 - \Delta^{-7/8})$..

Next, let $\gamma = \left(1 - \frac{1-\alpha}{(1+\alpha)k} \cdot (1 - \Delta^{-7/8})\right)$, we define the following random variable:

$$F_i = \frac{W_i}{\gamma^{i-1}}$$

Observe that F_i is a submartingale with a starting point $F_1 = W_1 = |N(u)|$. This is because for any $i \geqslant 2$:

$$\mathbb{E}[F_i \mid F_{i-1}] = \frac{1}{\gamma^{i-1}} \mathbb{E}[W_i \mid W_{i-1}] \leqslant \frac{W_{i-1}}{\gamma^{i-2}} = F_{i-1}$$

Hence, we would like to use Theorem 5.3.10 to get a concentration result for F_i and then deduce a concentration result for W_i. For this, we analyze $Var[F_i \mid F_{i-1}]$ and then $F_i - \mathbb{E}[F_i \mid F_{i-1}]$. Observe that:

$$\begin{aligned} Var[F_i \mid F_{i-1} \mid F_{i-1}] &= Var[F_i - F_{i-1}/\gamma \mid F_{i-1}] & (1) \\ &= \frac{1}{\gamma^{i-1}} Var[W_i - W_{i-1} \mid W_{i-1}] & (2) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{\gamma^{i-1}}\mathbb{E}[(W_i - W_{i-1})^2 \mid W_{i-1}] && (3)\\
&\leqslant \frac{1}{\gamma^{i-1}}\mathbb{E}[|W_i - W_{i-1}| \mid W_{i-1}] && (4)\\
&\leqslant \frac{1}{\gamma^{i-1}}\big(\mathbb{E}[W_{i-1} \mid W_{i-1}] - \mathbb{E}[W_i \mid W_{i-1}]\big) && (5)\\
&\leqslant \frac{1}{\gamma^{i-1}}\big(W_{i-1} - (1-8/k)W_{i-1}\big) && (6)\\
&\leqslant \frac{8W_{i-1}}{k\gamma^{i-1}} && (7)\\
&\leqslant \frac{16\Delta^{1.1}}{k} && (8)
\end{aligned}
$$

where (1) follows from Observation 5.3.5, (3) follows from the definition of Variance, (4) follows since the maximum value of $|W_i - W_{i-1}|$ is 1, (5) follows since $W_{i-1} \geqslant W_i$, (6) follows from Lemma 5.6.6 which implies that $\mathbb{E}[W_i \mid W_{i-1}] \geqslant (1-8/k)W_{i-1}$, and (8) follows since $W_i \leqslant W_1 \leqslant (1+\alpha)\Delta \leqslant 2\Delta$ and $\gamma^{i-1} \geqslant \Delta^{-0.1}$. Next, we analyze $F_i - \mathbb{E}[F_i \mid F_{i-1}]$.

$$
\begin{aligned}
F_i - \mathbb{E}[F_i \mid F_{i-1}] &= \frac{1}{\gamma^{i-1}}\big(W_i - \mathbb{E}[W_i \mid W_{i-1}]\big) && (1)\\
&\leqslant \frac{1}{\gamma^{i-1}}\big(W_{i-1} - (1-8/k)W_{i-1}\big) && (2)\\
&= \frac{8W_{i-1}}{k\gamma^{i-1}} && (3)\\
&\leqslant 112\Delta^{0.1} && (4)
\end{aligned}
$$

where (2) follows since by Lemma 5.6.6, $\mathbb{E}[W_i \mid W_{i-1}] \geqslant (1-8/k)W_{i-1}$, and (4) follows since $W_{i-1} \leqslant 2\Delta$, $k \geqslant \Delta/7$, and $\gamma^{i-1} \geqslant \Delta^{0.1}$. Therefore, by plugging the bounds on $Var[F_i \mid F_{i-1}]$ and $F_i - \mathbb{E}[F_i \mid F_{i-1}]$ into Theorem 5.3.10 for $\lambda = \Delta$, we get that:

$$
\begin{aligned}
&\mathbb{P}[F_t \geqslant (1+\alpha)\Delta + \lambda] \leqslant \mathbb{P}[F_t \geqslant F_1 + \lambda \mid \mathcal{B}]\\
&\leqslant \exp\left(-\frac{\lambda^2}{2\big((\sum_{i=1}^{t} 16\Delta^{1.1}/k) + 112\Delta^{0.1}\lambda\big)}\right)
\end{aligned}
$$

$$
\begin{aligned}
&= \exp\left(-\frac{\Delta^2}{2\left(16\Delta^{1.1}\log\Delta/100 + 224\Delta^{1.1}\right)}\right)\\
&= \exp-(\Delta^{0.8})
\end{aligned}
$$

where the first inequality holds since $F_1 \leqslant (1+\alpha)\Delta$. By a union bound over the event in which the event $\mathcal{B}$ doesn't happen we get that:

$$\mathbb{P}[F_t \geqslant 3\Delta] \leqslant \exp-(\Delta^{1/12})$$

Finally, since

$$
\begin{aligned}
W_t = F_t\cdot\gamma^{t-1} &\leqslant 3\Delta\cdot\left(1-\frac{1-\alpha}{(1+\alpha)k}\cdot(1-\Delta^{-7/8})\right)^{t-1}\\
&\leqslant 3\Delta\cdot\exp\left(-\frac{(k\log\Delta/100)(1-\Delta^{-7/8})}{3k}\right)\\
&\leqslant \Delta^{1-1/400}
\end{aligned}
$$

the claim follows. □

Now we are ready to give an upper bound on $|\mathcal{M}_u|$.

Lemma 5.6.12. *With probability at least* $1-\exp\left(-\Delta^{-1/15}\right)$ *it holds that:*

$$|\mathcal{M}_u| \leqslant \frac{\Delta}{2}\cdot\frac{(1+\alpha)^3}{(1-\alpha)^2}\cdot(1+\Delta^{-1/500})$$

Proof. Let $\mathcal{B}$ be the event in which the statement from Lemma 5.6.9 holds. That is, in the event $\mathcal{B}$ we have upper bounds on $|E_1|,\cdots,|E_t|$. As usual, we first prove the lemma conditioned on the event $\mathcal{B}$ happening. The proof is similar to the proof of Lemma 5.6.10 where we showed a lower bound on $|\mathcal{M}_u|$. Recall that Z_i is an indicator random variable indicating whether we add an edge to $\mathcal{M}_u$ in the ith iteration of the while loop in Algorithm 11, and $q_i = \mathbb{E}[Z_i \mid Z_1,\cdots,Z_{i-1}]$. In the proof of Lemma 5.6.10 we showed that $\sum_{i=1}^{t} q_i \leqslant \frac{\Delta}{2}\cdot\frac{(1+\alpha)^3}{(1-\alpha)^2}\cdot(1+\Delta^{-1/100})$. Let $|\mathcal{M}_u^t|$ be the size of $\mathcal{M}_u$ after the first t iterations were processed. We apply Theorem 5.3.11 with $X_i = Z_i$, $S_t = |\mathcal{M}_u^t|$, $p_i = q_i$, $M = 1$, $P = \frac{\Delta}{2}\cdot\frac{(1+\alpha)^3}{(1-\alpha)^2}\cdot(1+\Delta^{-1/100})$, and $\lambda = P + \Delta^{0.6}$ to get that:

$$\mathbb{P}[|\mathcal{M}_u^t| \geqslant \lambda \mid \mathcal{B}] \leqslant \exp\left(-\frac{\Delta^{1.2}}{8\Delta}\right) \leqslant \exp\left(-\Delta^{0.1}\right)$$

By a simple union bound argument (to get rid of the conditioning on $\mathcal{B}$), we get that $|\mathcal{M}_u^t| \leqslant P + \Delta^{0.6} \leqslant \frac{\Delta}{2} \cdot \frac{(1+\alpha)^3}{(1-\alpha)^2} \cdot (1 + \Delta^{-1/200})$ with probability at least $1 - \exp\left(-\Delta^{1/14}\right)$. Finally, by Lemma 5.6.11, with probability at least $1 - \exp\left(-\Delta^{1/12}\right)$, we can't add more than $\Delta^{0.9975}$ edges to $\mathcal{M}_u$ after the tth iteration. Hence, with probability at least $1 - \exp\left(-\Delta^{1/15}\right)$, it holds that $|\mathcal{M}_u| \leqslant \frac{\Delta}{2} \cdot \frac{(1+\alpha)^3}{(1-\alpha)^2} \cdot (1 + \Delta^{-1/500})$, as desired.

□

5.6.5 Finishing the Proof

Since $\mathcal{M}_u$ is exactly the matching adjacent to nodes in $N(u)$, the bounds on $|\mathcal{M}_u|$ suffice to prove Theorem 5.5.3.

Proof of Lemma 5.5.3. First, by Observation 5.6.2, Algorithm 10 and Algorithm 11 produce the same distributions over matchings in $(N(u) \times N^2(u)) \cap E$. Furthermore, by Lemma 5.6.12, with probability at least $1 - \exp\left(-\Delta^{1/15}\right)$, it holds that $|\mathcal{M}_u| \leqslant \frac{\Delta}{2} \cdot \frac{(1+\alpha)^3}{(1-\alpha)^2} \cdot (1 + \Delta^{-1/500})$. Hence, the number of unmatched neighbors of u is at least:

$$\begin{aligned}
deg'(u) &\geqslant (1-\alpha)\Delta - |\mathcal{M}_u| && (1)\\
&\geqslant (1-\alpha)\Delta - \frac{\Delta}{2} \cdot \frac{(1+\alpha)^3}{(1-\alpha)^2} \cdot (1+\Delta^{-1/500}) && (2)\\
&= \frac{\Delta}{2}\left(2 - 2\alpha - \frac{(1+\alpha)^3}{(1-\alpha)^2} \cdot (1+\Delta^{-1/500})\right) && (3)\\
&\geqslant \frac{\Delta}{2}\left(2 - 2\alpha - (1+8\alpha) \cdot (1+\Delta^{-1/500})\right) && (4)\\
&\geqslant \frac{\Delta}{2}\left(2 - 2\alpha - (1+8\alpha+\Delta^{-1/600})\right) && (5)\\
&= \frac{\Delta}{2}\left(1 - (10\alpha+\Delta^{-1/600})\right) && (6)
\end{aligned}$$

where (4) follows since $\frac{(1+\alpha)^3}{(1-\alpha)^2} \leqslant (1+8\alpha)$ for $\alpha \leqslant 1/10$, and rest follows from simple arithmetic. For the upper bound on $deg'(u)$, we use Lemma 5.6.10, which says that with probability at least $1 - \exp(-\Delta^{1/15})$, it holds that $|\mathcal{M}_u| \geqslant \frac{\Delta}{2} \cdot \frac{(1-\alpha)^3}{(1+\alpha)^2} \cdot (1 - \Delta^{-1/300})$. Hence, we have that:

$$\begin{aligned}
deg'(u) &\leqslant (1+\alpha)\Delta - |\mathcal{M}_u| && (1)\\
&\leqslant (1+\alpha)\Delta - \frac{\Delta}{2} \cdot \frac{(1-\alpha)^3}{(1+\alpha)^2} \cdot (1 - \Delta^{-1/300}) && (2)\\
&= \frac{\Delta}{2}\left(2 + 2\alpha - \frac{(1-\alpha)^3}{(1+\alpha)^2} \cdot (1 - \Delta^{-1/300})\right) && (3)\\
&\leqslant \frac{\Delta}{2}\left(2 + 2\alpha - (1 - 8\alpha) \cdot (1 - \Delta^{-1/300})\right) && (4)\\
&\leqslant \frac{\Delta}{2}\left(2 + 2\alpha - (1 - 8\alpha - \Delta^{-1/600})\right) && (5)\\
&\leqslant \frac{\Delta}{2}\left(1 + (10\alpha + \Delta^{-1/600})\right) && (6)
\end{aligned}$$

where (4) follows since $\frac{(1-\alpha)^3}{(1+\alpha)^2} \geqslant 1 - 8\alpha$, for $\alpha \leqslant 1/10$. Hence, by a simple union bound argument, we get the desired bounds on $deg'(u)$ with probability at least $1 - \exp(-\Delta^{1/16})$.

□

5.7 Lower Bounds

In this section, we complement our algorithms by giving lower bounds for bipartite regular graphs of low degree. Our bounds are based on a lower bound construction of Ben-Basat, Kawarabayashi, and Schwartzman (BKS), who proved $\Omega(1/\epsilon)$ lower bounds for a variety of problems in the LOCAL model, including maximum matching [71]. The high-level idea from that construction is as follows. First, design a symmetric subgraph gadget (for them, a path sufficed) with radius roughly equal to the number of rounds so that no matter how the gadget's boundary nodes are hooked up to the surrounding graph, the innermost node(s) can never hope to learn about anything outside of the gadget. Next, due to the symmetrical design of the gadget, we can choose to randomly insert it either forwards or backwards. No matter how an algorithm (even randomized) originally chooses to match that innermost gadget node(s), this equalizes the probability of matching it with its forward neighbor and backwards neighbor. Finally, there is now a constant probability that a constant number of these innermost matching edges are now locked in a way that induces a single error, meaning we expect there to be roughly one error per (gadget size) number of nodes. Against Monte Carlo algorithms, there is an additional step where the independence of our forwards/backwards insertion decisions can be used to execute a Chernoff bound that makes it exponentially unlikely that the number of mistakes can be significantly lower than expected.

The efficiency of this argument hinges on the number of nodes in the gadget. Let r be the number of rounds available to the algorithm. In the original construction [71], the path gadgets had $O(r)$ nodes, so roughly ϵ^{-1} rounds winds up translating into a $(1+\epsilon)$ multiplicative error. Our main technical contribution is showing that we can match this efficiency when constructing a cycle (a bipartite regular graph of degree two) and that as the degree Δ scales up, we can design gadgets with only $O(\Delta r)$ nodes. We make the following observations about this dependence on Δ: (i) any lower bound, whether it uses this particular gadget framework or not, must eventually worsen as Δ increases because our upper bounds establish that higher degree regular graphs are easy and (ii) for this particular gadget framework, $\Theta(\Delta r)$ is asymptotically the best possible gadget size dependence on Δ and r. The latter can be observed by considering a breadth-first search (BFS) tree rooted at one of the innermost nodes; this tree must have r levels before we ever reach any node outside the gadget. There must be a path from the root to a node at level r, otherwise the root will not be connected to the outside graph; this path is a witness to the fact that there is at least one node at every level. Next, observe that the node at level $i \in [1, r]$ must have Δ unique neighbors, which must be either in levels $i-1$, i, or $i+1$ (otherwise the BFS tree is incorrect); this holds even for non-bipartite graphs and for bipartite graphs they can only be in levels $i-1$ or $i+1$. In any case, this means we can partition the r levels into groups of three (if non-bipartite) or two (if bipartite) such that each group must have at least Δ nodes. Hence there are $\Omega(\Delta r)$ nodes in such a gadget.

Our gadgets formally yield the following lower bounds against deterministic and Monte Carlo algorithms.

Theorem 5.7.1. *For any degree $\Delta \geqslant 2$ and error $\epsilon \in (0, \frac{1}{160\Delta+32})$, any deterministic LOCAL algorithm that computes a $(1+\epsilon)$-multiplicative approximation for* MAXIMUM-MATCHING *on bipartite Δ-regular graphs with at least $n \geqslant \Omega(\Delta^{-1}\epsilon^{-1})$ nodes requires*

$\Omega(\Delta^{-1}\epsilon^{-1})$ *rounds.*

For any degree $\Delta \geqslant 2$*, error* $\epsilon \in (0, \frac{1}{320\Delta+64})$*, and failure probability* $\delta \in (0,1)$*, any Monte Carlo LOCAL algorithm that computes a* $(1+\epsilon)$*-multiplicative approximation with probability at least* $1-\delta$ *for* MAXIMUMMATCHING *on bipartite* Δ*-regular graphs with at least* $n \geqslant \Omega(\Delta^{-1}\epsilon^{-1}\ln(1-\delta)^{-1})$ *nodes requires* $\Omega(\Delta^{-1}\epsilon^{-1})$ *rounds.*

The complexity of our proof and the resulting constant factors heavily depend on the degree, so we will split our analysis and gadget design into the following cases, from easiest to hardest: degree two (Section 5.7.1), even degree (Section 5.7.2), and finally general degree (Section 5.7.3). The first two subsections prove specializations of Theorem 5.7.1 with better bounds.

5.7.1 Bipartite Regular Graphs of Degree Two (Cycles)

As a warm-up, we consider the (easy difficulty) degree two case. This case is also covered by Section 5.7.2, which handles even degree, but the construction here will be simpler, be useful for building intuition (especially for readers less familiar with the original BKS construction), and yield a better constant factors. We will prove the following subresult in this subsection.

Theorem 5.7.2. *For any error* $\epsilon \in (0, \frac{1}{16})$*, any deterministic LOCAL algorithm that computes a* $(1+\epsilon)$*-multiplicative approximation for* MAXIMUMMATCHING *on bipartite 2-regular graphs with at least* $n \geqslant \Omega(\epsilon^{-1})$ *nodes requires* $\Omega(\epsilon^{-1})$ *rounds.*

For any error $\epsilon \in (0, \frac{1}{32})$*, and failure probability* $\delta \in (0,1)$*, any Monte Carlo LOCAL algorithm that computes a* $(1+\epsilon)$*-multiplicative approximation with probability at least* $1-\delta$ *for* MAXIMUMMATCHING *on bipartite 2-regular graphs with at least* $n \geqslant \Omega(\epsilon^{-1}\ln(1-\delta)^{-1})$ *nodes requires* $\Omega(\epsilon^{-1})$ *rounds.*

Proof. The overall proof plan is to use the BKS path gadget (with slightly different notation), but embed it a little differently to get an even-length cycle. We give all the proof details here so it is not necessary for a reader to be familiar with the BKS proof.

We want our gadget to help guard against LOCAL algorithms which run for some r rounds (which will be one round fewer than the $\Omega(\epsilon^{-1})$ rounds we require in the theorem statement). Figure 5.2 depicts the gadget design, which we now formally describe. For the ith copy of our (path) gadget, let us label the nodes $v_0^i, v_1^i, \ldots, v_{2r}^i$ and connect v_j^i with v_{j+1}^i for $j \in \{0, 1, \ldots, 2r-1\}$. Our gadget will

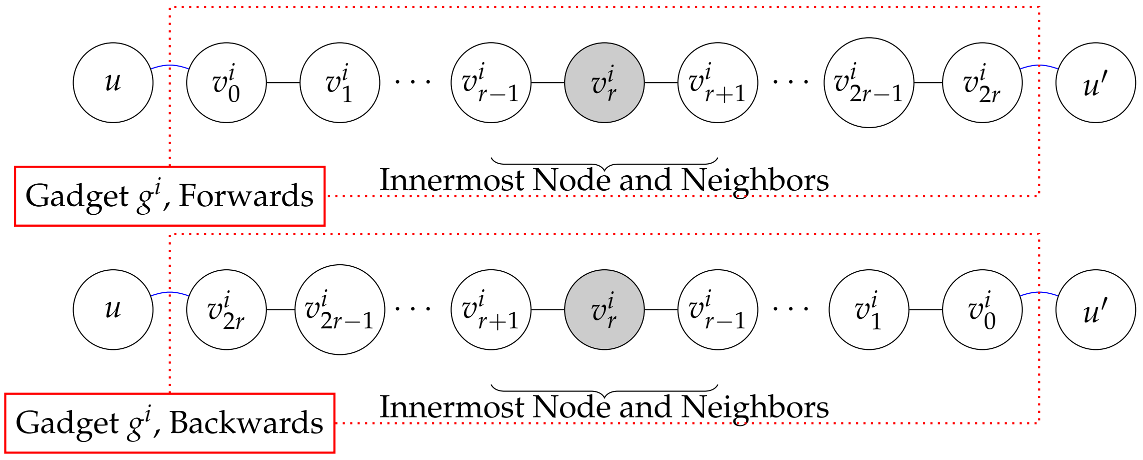

Figure 5.2: (Degree Two Case) These are the two ways to our (path) gadget may be inserted between u and u'. In r rounds, the shaded innermost node v_r^i can only learn about the internal contents of the gadget and nothing else in the surrounding graph, no matter whether the gadget was inserted forwards or backwards. The gadget is connected to two anchor nodes via the blue curved edges.

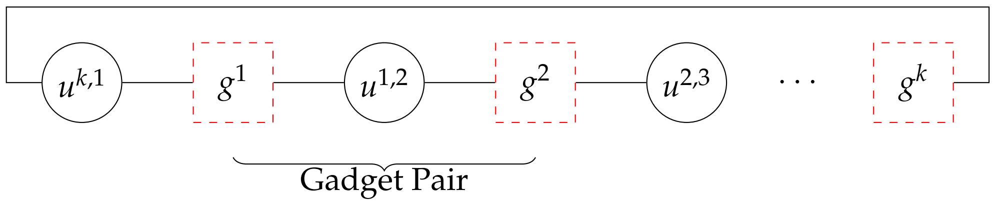

Figure 5.3: (Degree Two Case) For even k, we generate a distribution of counterexamples by inserting k gadgets between k anchor nodes $u^{1,2}, u^{2,3}, \cdots, u^{k-1,k}, u^{k,1}$. Each gadget is independently and uniformly at random chosen to be inserted forwards or backwards. We then pair up gadgets, starting with $\{g^1, g^2\}$.

also have two external edges coming out of the path endpoints v_0^i and v_{2r}^i. These will connect to some external nodes u, u'. Let's say that the gadget is inserted forwards if v_0^i is connected to u and v_{2r}^i is connected to v and the gadget is inserted backwards if v_{2r}^i is connected to u and v_0^i is connected to v. Importantly, for the innermost node v_r^i, it is impossible to tell whether the gadget is inserted forwards or backwards within r rounds.

Now that we have described our gadget, we are ready to discuss how to use it. Let k be the number of gadgets that we use; since we will pair them up later, we know that k is even. A constant number of gadgets will suffice against deterministic LOCAL algorithms, but we will require on the order of $\ln(1-\delta)^{-1}$ gadgets against Monte Carlo LOCAL algorithms to establish an failure probability of δ. We introduce k additional anchor nodes, outside of any gadget, and label them $u^{1,2}, u^{2,3}, \ldots, u^{k-1,k}, u^{k,1}$, with the intention that anchor node $u^{i,j}$ is connected to gadgets g^i and g^j. We then insert each gadget independently and uniformly at random either forwards or backwards between its two anchor nodes. The final result is a distribution over (counterexample) graphs; see Figure 5.3 for a depiction.

As a preliminary observation, each graph in the support of distribution is a cycle on $k(2r+2)$ nodes. Since this is an even number of nodes, the graphs in our distribution are indeed bipartite.

We are now ready to begin reasoning about the algorithm's performance on this distribution. Let us focus our attention on some innermost gadget node v_r^i (note that each gadget pair has two of these). The algorithm may do one of the following to this gadget node (a deterministic LOCAL algorithm does one of them deterministically, and a Monte Carlo LOCAL algorithm will produce some distribution over these three possibilities):

- leave the node v_r^i unmatched,
- match v_r^i with v_{r-1}^i,
- or match v_r^i with v_{r+1}^i.

For technical reasons, we will post-process the algorithm's output to make the first case impossible. Whenever v_r^i is unmatched, we will match it with v_{r+1}^i, unmatching v_{r+1}^i with its previous match. This can only improve the size of the matching, so if we can show that the post-processed matching is not a $(1+\epsilon)$-approximation, the algorithm's matching is also not a $(1+\epsilon)$-approximation.

Next, we say that v_r^i is matched to the right if it is matched with v_{r+1}^i and its gadget is inserted forwards, or if it is matched with v_{r-1}^i and its gadget is inserted backwards. Similarly, we say that it is matched to the left if it is matched with v_{r-1}^i and its gadget is inserted forwards, or if it is matched with v_{r+1}^i and its gadget is inserted backwards. Our post-processing has guaranteed that exactly one of these possibilities occurs, and our random insertions guarantee each innermost node is independently and uniformly at random matched to the right or left (we have essentially XOR'd the algorithm's decision pattern with a random bit string). Interestingly, this means that we can even guard against a Monte Carlo algorithm whose computation at various nodes all have access to some shared public randomness.

We now pair up adjacent gadgets; each pair will have an independent chance of creating an unmatched node. For $i \in \{1, 2, ..., k/2\}$, the ith gadget pair includes the nodes of g^{2i-1}, the nodes of g^{2i}, as well as the anchor node between them, namely $u^{2i-1,2i}$. We know that there is a $1/2$ probability that the two innermost gadget nodes v_r^{2i-1} and v_r^{2i} have one match left and one match right. When this event occurs, there are an odd number of nodes between these two matching edges, so one of the nodes between these two innermost gadget nodes must be unmatched. Furthermore, this $1/2$ failure chance occurs independently across all our gadget pairs.

Against deterministic LOCAL algorithms, we are essentially done. We now choose the minimum number of gadgets possible: $k = 2$. We also choose $r = \lfloor \frac{1}{9}(\epsilon^{-1} - 7) \rfloor$ which is $\Omega(\epsilon^{-1})$; since $\epsilon < 1/16$ this guarantees $r \geqslant 1$ and hence v_{r-1}^i and v_{r+1}^i actually exist. Furthermore, our overall construction has $n = k(2r + 2) = 4r + 4 = \Omega(\epsilon^{-1})$ nodes, which the algorithm was guaranteed to be able to handle. Observe that in expectation (over the randomness of our counterexample distribution) the algorithm leaves half a node unmatched across these two gadgets. It must get performance at least this bad on some specific graph in the support of this distribution. On that counterexample, its multiplicative approximation is at least:

$$\begin{aligned}
\frac{n}{n - 1/2} &= 1 + \frac{1}{2n - 1} \\
&= 1 + \frac{1}{8r + 7} \\
&= 1 + \frac{1}{8\lfloor \frac{1}{9}(\epsilon^{-1} - 7) \rfloor + 7} \\
&\geqslant 1 + \frac{1}{\frac{8}{9}(\epsilon^{-1} - 7) + 7} \\
&> 1 + \epsilon && (\text{since } \epsilon < 1/16 \implies \epsilon < 1/7)
\end{aligned}$$

This contradicts the $(1 + \epsilon)$-multiplicative approximation guarantee of the deterministic LOCAL algorithm, completing that half of the proof.

We now consider Monte Carlo LOCAL algorithms. We now need additional gadgets to force the failure probability to δ, so we will be choosing a new value for k. Since each gadget pair is independent, we can use a Chernoff bound to control the probability that the algorithm fails on too few gadget pairs; it will be exponentially small in the total number of gadgets.

Due to linearity of expectation, the expected number of gadget pairs which fail is $\mu = \frac{k}{2} \cdot \frac{1}{2} = \frac{k}{4}$. We need some additional multiplicative error to run our Chernoff bound argument, so let us consider the likelihood that at most half as many, $\frac{k}{8}$ fail. Let X be a random variable for the total number of gadget pair failures; since X is a sum of independent Bernoulli variables we know that:

$$\begin{aligned}\mathbf{Pr}[X \leqslant (1-\gamma)\mu] &\leqslant e^{-\gamma^2\mu/2} \qquad \forall \gamma \in (0,1)\\ \mathbf{Pr}[X \leqslant k/8] &\leqslant e^{-(1/2)^2(k/4)/2}\\ \mathbf{Pr}[X \leqslant k/8] &\leqslant e^{-k/32}\end{aligned}$$

To arrive at a contradiction, we want to show that the left-hand side is strictly less than $1-\delta$. Hence it suffices to show that:

$$\begin{aligned}e^{-k/32} &< 1-\delta\\ -k/32 &< \ln(1-\delta)\\ k &> 32\ln(1-\delta)^{-1}\end{aligned}$$

We can easily satisfy this by choosing $k = 2\lceil 16\ln(1-\delta)^{-1}\rceil + 2$; note that this is guaranteed to be even and will contribute an asymptotic factor of $\Omega(\ln(1-\delta)^{-1})$. Since we have now forced $\mathbf{Pr}[X \leqslant k/8] < 1-\delta$, we know that the complement probability is $\mathbf{Pr}[X > k/8] > \delta$. When this many failures occur, our Monte Carlo algorithm will have an approximation ratio which is strictly larger than:

$$\begin{aligned}\frac{n}{n-k/8} &= 1 - \frac{k/8}{n-k/8}\\ &= 1 - \frac{k/8}{k(2r+2)-k/8}\\ &= 1 - \frac{1}{16r+15}\end{aligned}$$

Similar to the deterministic case, we can choose $r = \lfloor \frac{1}{17}(\epsilon^{-1} - 15)\rfloor$ which is $\Omega(\epsilon^{-1})$ and since $\epsilon < 1/32$ this guarantees $r \geqslant 1$ which ensures v^i_{r-1} and v^i_{r+1} exist. Additionally our construction has $n = k(2r+2) = \Omega(\epsilon^{-1}\ln(1-\delta)^{-1})$ nodes which the algorithm was guaranteed to be able to handle. We now finish the approximation ratio analysis; the ratio is strictly larger than:

$$\begin{aligned}\frac{n}{n-k/8} &= 1 - \frac{1}{16\lfloor\frac{1}{17}(\epsilon^{-1}-15)\rfloor + 15} \\ &\geqslant 1 - \frac{1}{\frac{16}{17}(\epsilon^{-1}-15)+15} \\ &> 1+\epsilon \qquad\qquad (\text{since } \epsilon < 1/32 \implies \epsilon < 1/15)\end{aligned}$$

Hence with probability strictly greater than δ, the approximation ratio is strictly worse than $(1+\epsilon)$. This contradicts the approximation guarantee of the Monte Carlo LOCAL algorithm, completing the second part of the proof. □

Remark 5.7.3. In addition to being able to guard against Monte Carlo algorithms with access to shared public randomness, it also does not matter if the algorithms know the graph size n upfront. It is also possible to improve the upper bounds on allowed ϵ with additional observations, e.g. removing anchor nodes or arguing that in the deterministic case at least two nodes must actually be unmatched. At some point, this hinges on what a LOCAL algorithm is actually allowed to do in zero rounds.

5.7.2 Bipartite Regular Graphs of Even Degree

We now consider the (medium difficulty) even degree case. We prove the following subresult in this subsection.

Theorem 5.7.4. *For any even degree $\Delta \geqslant 2$ and error $\epsilon \in (0, \frac{1}{16\Delta+9})$, any deterministic LOCAL algorithm that computes a $(1+\epsilon)$-multiplicative approximation for* MAXIMUMMATCHING *on bipartite Δ-regular graphs with at least $n \geqslant \Omega(\Delta^{-1}\epsilon^{-1})$ nodes requires $\Omega(\Delta^{-1}\epsilon^{-1})$ rounds.*

For any even degree $\Delta \geqslant 2$, error $\epsilon \in (0, \frac{1}{32\Delta+17})$, and failure probability $\delta \in (0,1)$, any Monte Carlo LOCAL algorithm that computes a $(1+\epsilon)$-multiplicative approximation with probability at least $1-\delta$ for MAXIMUMMATCHING *on bipartite Δ-regular graphs with at least $n \geqslant \Omega(\Delta^{-1}\epsilon^{-1}\ln(1-\delta)^{-1})$ nodes requires $\Omega(\Delta^{-1}\epsilon^{-1})$ rounds.*

Proof. In order to support larger degree than the degree two proof, we plan to replace each node v from that construction with a "node subgadget" h^v. Just as each node used to be connected to two other nodes, we will be able to connect each node subgadget to two other node subgadgets.

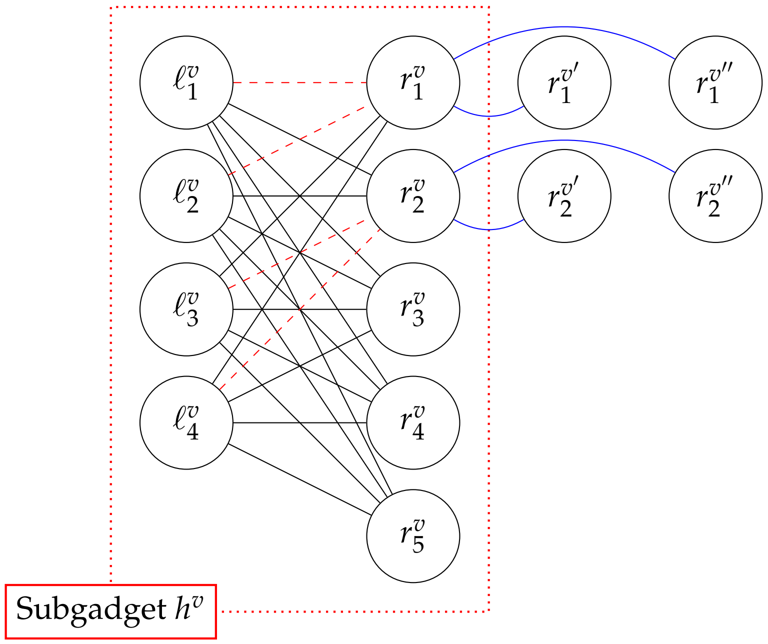

Figure 5.4: (Even Degree Case) This is our node subgadget h^v for degree $\Delta = 4$, which takes on the role of a node v from the original construction. The red dashed lines emphasize missing edges (if they were not missing, the gadget would internally look like biclique $K_{\Delta,\Delta+1}$). The subgadget is connected to two other subgadgets $h^{v'}, h^{v''}$ via blue curved edges to correspond with the original node being connected to two other nodes v' and v''. To maintain the bipartiteness of the resulting graph, the nodes being replaced by these node subgadgets must all be from *even-length* cycles.

Informally, our subgadget will offer $\Delta/2$ nodes that are each missing two degree and hence can be connected to adjacent node subgadgets, for a total of Δ outgoing edges. We will then expect the typical solution to use exactly one of these external edges, which simulates the behavior of a node in the original proof. The fact that we have Δ outgoing edges is not a coincidence; a combination of regularity and bipartiteness mean that if we want all subgadget nodes with external edges to be on the same side of the bipartition, the number of external edges is necessarily a multiple of Δ (since each node on that side generates Δ edges and each node on the other side consumes Δ edges).

Figure 5.4 depicts the subgadget design for degree $\Delta = 4$, and we now formally describe it for all even $\Delta \geqslant 2$. To replace what was originally node v, our (node) subgadget consists of the $(2\Delta + 2)$ nodes $\ell_1^v, \ell_2^v, ..., \ell_\Delta^v, r_1^v, r_2^v, ..., r_{\Delta+1}^v$. Internally, we connect $\ell_{j'}^v$ with r_j^v for all for all $j' \in \{1, 2, ..., \Delta\}$ and $j \in \{1, 2, ..., \Delta + 1\}$ unless $j' \in \{2j - 1, 2j\}$. Our gadget will also have Δ external edges coming out of nodes $r_1^v, r_2^v, ..., r_{\Delta/2}^v$ (two each). If the node v was connected to nodes v' and v'', then we have external edges from r_j^v to $r_j^{v'}$ and $r_j^{v''}$ for all $j \in \{1, 2, ..., \Delta/2\}$.

Now that we have described our subgadget, we copy the path gadget construction from the proof of Theorem 5.7.2, replacing every node with a node subgadget. Reusing some notation from that proof, our path gadgets will guard against algorithms that run for r rounds and we will be using k path gadgets in total, where k is even and to be decided later. We also use k additional anchor nodes which also get turned into node subgadgets. In total, the total number of nodes we use is now $k(2r + 2)(2\Delta + 1)$.

Let us consider what the algorithm does on a single node subgadget. For convenience, let's define the following sets of subgadget v's nodes:

$$\begin{aligned} X^v &= \{\ell_1^v, ..., \ell_\Delta^v\} && (|X^v| = \Delta) \\ Y^v &= \{r_1^v, ..., r_{\Delta/2}^v\} && (|Y^v| = \Delta/2) \\ Z^v &= \{r_{\Delta/2+1}^v, ..., r_{\Delta+1}^v\} && (|Z^v| = \Delta/2 + 1) \end{aligned}$$

The most natural thing to do for subgadget v is to match the $\Delta/2 + 1$ nodes of Z^v to an equal number of nodes in X^v, then match the remaining nodes of X^v to an equal number of nodes in Y^v. This leaves one remaining node from Y^v to match with an external node.

In fact, we will force the algorithm to do this on each node subgadget $h^{v_r^i}$ that corresponds to an innermost node v_r^i in some path gadget g^i. For convenience, let $v = v_r^i$. We will postprocess the algorithm's decisions on subgadget h^v, and we will maintain that the total size of the matching does not decrease, so if the post-processed matching fails the approximation guarantee, so did the algorithm's original matching. First, if any node in Z^v is unmatched, we match it to a node in X^v which is not currently matched to another node in Z^v. This is always possible because Z^v and X^v are fully connected and $|X^v| = \Delta \geqslant \Delta/2 + 1 = |Z^v|$. We possibly need to unmatch that node of Z^v with a node in Y^v, but even with this in mind, we have not reduced the total size of the matching. Next, if any node in X^v is unmatched after this, we match it to a node in Y^v which is not currently matched to another node in Y^v. This is always possible because X^v and Y^v are almost fully connected; each node in X^v is only missing a connection to a single node in Y^v, and between Y^v and Z^v there are $\Delta + 1$ nodes total, enough to guarantee a match for every node in X^v even with the missing edges. We possibly need to unmatch that node of Y^v with an external node, but again this does not reduce the total size of the matching. Finally, if any node in Y^v is unmatched after this, we match it to an external node in $h^{v^i_{r+1}}$, possibly unmatching that external node with a different external node. Since we are only doing this post-processing for innermost nodes $v = v^i_r$, we can safely do this all without collisions. As a result of our post-processing, exactly one node of $h^{v^i_r}$ is matched to an external node, and the size of the matching has not decreased.

At this point, we can follow the original proof with only minor changes. We say that $h^{v^i_r}$ is matched to the right if its only external edge matched is with $h^{v^i_{r+1}}$ and its gadget g^i is inserted forwards, or if its only external edge matched is with $h^{v^i_{r-1}}$ and its gadget g^i is inserted backwards. Similarly, we say that it is matched to the left if its only external edge edge matched is with $h^{v^i_{r-1}}$ and its gadget g^i is inserted forwards, or if its only external edge matched is with $h^{v^i_{r+1}}$ and its gadget g^i is inserted backwards. Again, our post-processing has guaranteed that exactly one of these possibilities occurs, and our random insertions guarantee each innermost node subgadget is independently an uniformly at random matched to the right or left.

We now pair up adjacent gadgets as before; for $i \in \{1, 2, ..., k/2\}$, the ith gadget pair includes the node subgadgets of g^{2i-1}, the node subgadgets of g^{2i}, as well as the node subgadget for the anchor node $u^{2i-1,2i}$. We know that there is a $1/2$ probability that the two innermost node subgadgets have one match left and one match right. When this event occurs, there are an odd number of nodes between these two matching edges (an odd number of subgadgets times an odd number of nodes per subgadget), so one of these nodes must be unmatched. Furthermore, this $1/2$ failure chance occurs independently across all our gadget pairs.

Against deterministic LOCAL algorithms we again choose a minimal $k = 2$ for a single gadget pair. On average against our distribution, the algorithm leaves $1/2$ of a node unmatched; we can begin to work out its multiplicative approximation ratio to be at least:

$$\begin{aligned}\frac{n}{n - 1/2} &= 1 + \frac{1}{2n-1} \\ &> 1 + \frac{1}{2n}\end{aligned}$$

$$= 1 + \frac{1}{2k(2r+2)(2\Delta+1)}$$
$$= 1 + \frac{1}{8(2\Delta+1)(r+1)}$$

To turn this into $(1+\epsilon)$, we let $r = \lfloor \frac{\epsilon^{-1}-1}{16\Delta+8} \rfloor$. This is $\Omega(\Delta^{-1}\epsilon^{-1})$ as claimed, and as long as $\epsilon < \frac{1}{16\Delta+9}$, we have $r \geqslant 1$ so that our post-processing was valid (i.e. v^i_{r-1} and v^i_{r+1} exist). Additionally our construction has $\Omega(\Delta^{-1}\epsilon^{-1})$ nodes which the algorithm was guaranteed to be able to handle. The upshot is that after making these choices for k and r, we have contradicted the $(1+\epsilon)$-multiplicative approximation guarantee of the deterministic LOCAL algorithm, completing that half of the proof.

To deal with the other half, Monte Carlo LOCAL algorithms, we again invoke a Chernoff bound. Since the failure probability is the same, the Chernoff calculation is identical and we need to choose $k = 2\lceil 16 \ln(1-\delta)^{-1} \rceil + 2$ so that the number of gadget pair failures X satisfies $\mathbf{Pr}[X \leqslant k/8] < 1 - \delta$. Hence the complement probability is $\mathbf{Pr}[X > k/8] < \delta$, but this many failures would make our Monte Carlo algorithm have an approximation which is strictly larger than:

$$\frac{n}{n-k/8} = 1 + \frac{k/8}{n-k/8}$$
$$= 1 + \frac{k/8}{k(2r+2)(2\Delta+1) - k/8}$$
$$> 1 + \frac{1}{16(2\Delta+1)(r+1)}$$

To turn this into $(1+\epsilon)$, we let $r = \lfloor \frac{\epsilon^{-1}-1}{32\Delta+16} \rfloor$. This is $\Omega(\Delta^{-1}\epsilon^{-1})$ as claimed, and as long as $\epsilon < \frac{1}{32\Delta+17}$, we have $r \geqslant 1$ so that our post-processing was valid. Additionally our construction has $\Omega(\Delta^{-1}\epsilon^{-1}\ln(1-\delta)^{-1})$ nodes which the algorithm was guaranteed to be able to handle. Hence with probability strictly greater than δ, the approximation ratio is strictly worse than $(1+\epsilon)$. This contradicts the approximation guarantee of the Monte Carlo LOCAL algorithm, completing the second part of the proof. □

5.7.3 Bipartite Regular Graphs of General Degree

We now consider the (hard difficulty) general degree case. Unfortunately, the way we got the even degree proof to work via node subgadgets runs into complications when try the same technique for odd degree. If we design a node subgadget with an odd number of nodes, we will have an odd number of external edges and the algorithm will be able to break symmetry (the subgadget's two neighbors will not look the same). If we design a node subgadget with an even number of nodes, then we will want to match an even number of nodes to external nodes and the algorithm will be able to evenly split them between the subgadget's two neighbors.

Hence we will need a different attack angle to handle odd degrees. The key insight is to view the path gadget as a degree-two way to simulate a very long edge (so that the algorithm cannot see the neighbor on the other side of the long edge). We will design edge subgadgets to do the same thing for higher degrees.

We are now ready to prove Theorem 5.7.1, which is restated below for convenience.

Theorem 5.7.1. *For any degree $\Delta \geqslant 2$ and error $\epsilon \in (0, \frac{1}{160\Delta+32})$, any deterministic LOCAL algorithm that computes a $(1+\epsilon)$-multiplicative approximation for* MAXIMUMMATCHING *on bipartite Δ-regular graphs with at least $n \geqslant \Omega(\Delta^{-1}\epsilon^{-1})$ nodes requires $\Omega(\Delta^{-1}\epsilon^{-1})$ rounds.*

For any degree $\Delta \geqslant 2$, error $\epsilon \in (0, \frac{1}{320\Delta+64})$, and failure probability $\delta \in (0,1)$, any Monte Carlo LOCAL algorithm that computes a $(1+\epsilon)$-multiplicative approximation with probability at least $1-\delta$ for MAXIMUMMATCHING *on bipartite Δ-regular graphs with at least $n \geqslant \Omega(\Delta^{-1}\epsilon^{-1}\ln(1-\delta)^{-1})$ nodes requires $\Omega(\Delta^{-1}\epsilon^{-1})$ rounds.*

Proof. In order to support larger degree than the degree two proof, we plan to create a (bipartite, regular) degree five graph and then replace each edge (u,v) in it with an "edge subgadget" $h^{u,v}$. We choose degree five to be the original degree because of two factors: we want to write every degree $\Delta \geqslant 2$ as a sum $\Delta = 3x + 2y$ where $x, y \in \{0, 1, ..., \Delta\}$ and we needed the coefficients on x and y in that expression to both be at least two. It is easy to prove that every degree $\Delta \geqslant 2$ fits this criteria; if Δ is even, then it can be written as $\Delta = 0 + 2y$ for some $y \geqslant 1$ and we can then always choose $x = 0$. If Δ is odd, then it is at least $\Delta \geqslant 3$ and so $\Delta - 3$ is even; we can hence write $\Delta = 3 + 2y$ for some $y \geqslant 0$ and then always choose $x = 1$.

This property that $\Delta = 3x + 2y$ means that for any specific value of Δ we will only use two different edge subgadgets. Our "blue" subgadget will induce degree x and our "red" subgadget will induce degree y, and hence three blue subgadgets and two blue subgadgets induce a real degree of Δ.

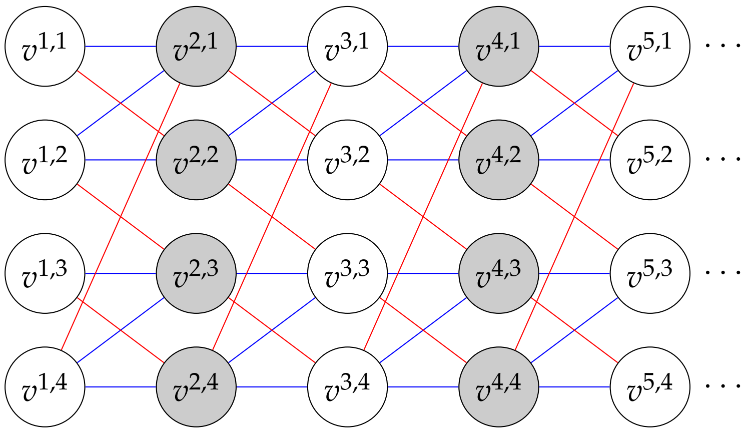

Figure 5.5: (General Degree Case) These are nodes from five adjacent layers of our base graph, which is bipartite, regular, and has degree five. The edges are colored blue and red so that every node has three incident blue edges and two incident red edges. Each of these colored edges (u, v) will be replaced by an edge subgadget $h^{u,v}$.

The exact degree five base graph is not actually important (we can always extract five disjoint perfect matchings and arbitrarily color them three blue, two red), but for concreteness we will reason about the following layered graph. Let k, the number of layers, be a parameter chosen later; we guarantee that k will be a multiple of four (each set of four layers corresponds to a gadget pair in the original proof). We have four nodes per layer, so our graph consists of $4k$ nodes labelled $v^{i,j}$ for $i \in \{1, 2, ..., k\}$ and $j \in \{1, 2, 3, 4\}$. For all $i, i' \in \{1, 2, ..., k\}$, where $i + 1 \equiv i' \pmod{k}$, and for all $(j, j') \in \{(1, 1), (2, 1), (2, 2), (3, 3), (4, 3), (4, 4)\}$ we have an blue edge $(v^{i,j}, v^{i',j'})$. For all $i, i' \in \{1, 2, ..., k\}$, where $i + 1 \equiv i' \pmod{k}$, and for all $(j, j') \in \{(1, 2), (2, 3), (3, 4), (4, 1)\}$ we have an red edge $(v^{i,j}, v^{i',j'})$. This base graph is depicted in Figure 5.5.

We are now ready to present our edge subgadget $h^{u,v}$. It takes in three parameters: the degree Δ, an induced degree $z \in \{0, 1, ..., \Delta\}$, and a length ρ which is a positive even integer. Informally, our edge subgadget behaves like z parallel edges between u and v that have a length of $2\rho + 1$ (i.e. it takes that many hops to get from u to v), implying that a 2ρ-round algorithm cannot see the identity of v when making a matching decision for u. This is done by adding $\Delta\rho$ additional nodes along with edges that use up all of their Δ degree along with using up z degree from both u and v.

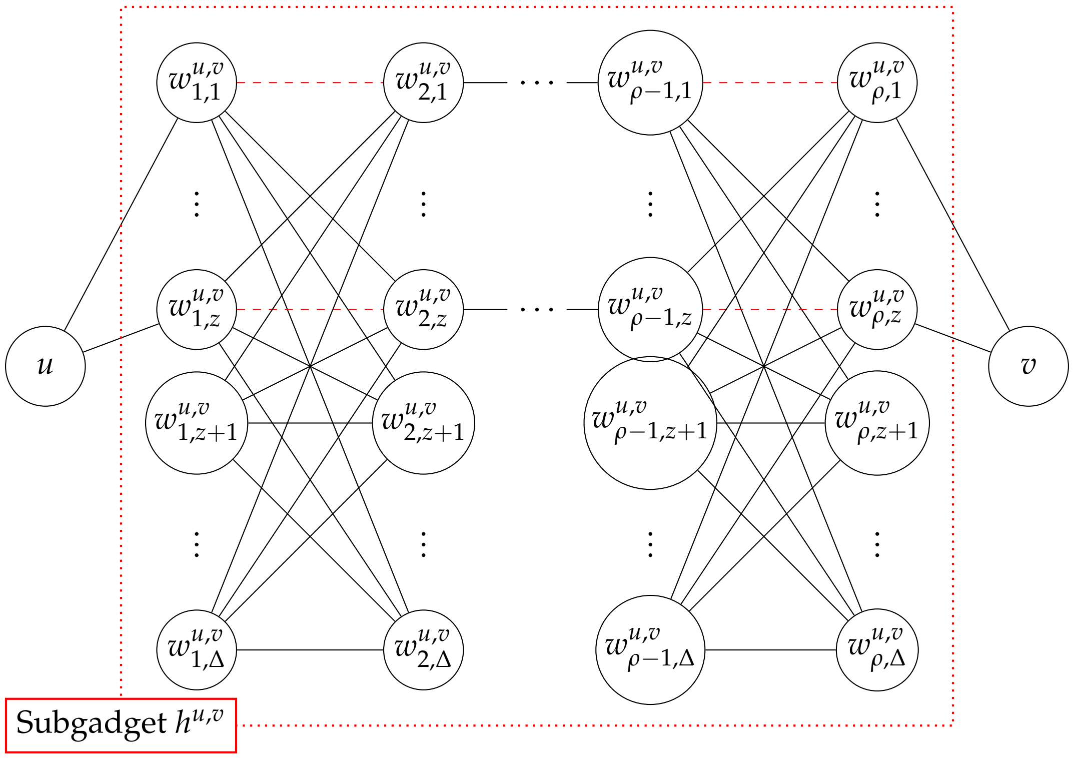

Figure 5.6: (General Degree Case) This is our edge subgadget $h^{u,v}$, which simulates z parallel edges between u and v of length $2\rho + 1$. The red dashed lines emphasize missing edges. To maintain the bipartiteness of the resulting graph, u and v must be on opposite sides of the bipartition.

We now formally describe the edge subgadget, which is depicted in Figure 5.6. Our subgadget consists of $(\Delta\rho)$ nodes $\{w^{u,v}_{i,j}\}_{i\in\{1,2,...,\rho\},j\in\{1,2,...,\Delta\}}$. We think of our nodes as being arranged in ρ layers with Δ nodes in each layer. We say that a node $w^{u,v}_{i,j}$ is in layer i, and each layer is only connected to adjacent layers (except for the first layer $i = 1$ and last layer $i = \rho$, which are also connected to u and v respectively).

When going up from an odd layer $i \in \{1, 3, ..., \rho - 1\}$ to the even layer $i+1$, the nodes are fully connected except for a matching between the first z nodes. That is, $w^{u,v}_{i,j}$ is connected to $w^{u,v}_{i+1,j'}$ for all $j, j' \in \{1, 2, ..., \Delta\}$ unless $j = j' \leqslant z$. When going up from an even layer $i \in \{2, 4, ..., \rho - 2\}$ to the even layer $i + 1$, the only connection is a matching between the first z nodes. That is, $w^{u,v}_{i,j}$ is connected to $w^{u,v}_{i+1,j}$ for all $j \in \{1, 2, ..., z\}$. Finally, u is connected to $w^{u,v}_{1,j}$ for $j \in \{1, 2, ..., z\}$ and v

is connected to $w^{u,v}_{\rho,j}$ for $j \in \{1, 2, ..., z\}$.

After replacing all the edges in our base graph with edge subgadgets, we have a total of $n = 4k + 10k(\Delta\rho)$ nodes. Next, we will identify the equivalents of path gadgets and gadget pairs within our graph, so that we can argue that algorithms will wind up making mistakes at a certain rate.

Our new version of a gadget consists of an innermost node $v^{i,j}$ with *even* i along with the five edge subgadgets incident to that node. Our construction has provided the following properties: (i) gadgets do not overlap and (ii) each gadget can be removed and reinserted into the graph one of $3!2! = 12$ ways without changing the overall structure of the graph, since the blue edge subgadgets can be freely permuted with each other and the red edge subgadgets can be freely permuted with each other. This leaves all nodes $v^{i,j}$ with *odd* i serving as anchor nodes between gadgets.

We now group gadgets as follows; for $i \in \{1, 2, ..., k/4\}$, the ith gadget group includes the 5 gadgets whose innermost nodes are $v^{4i-2,1}, v^{4i-2,2}, v^{4i-2,4}, v^{4i,1}, v^{4i,2}$ (each consisting of that node and the five edge subgadgets incident to that node), as well as the anchor node $v^{4i-1,1}$. Intuitively, the situation is that we are examining an anchor node and its five neighbors; the algorithm needs exactly one of these neighbors to be matched towards our anchor node, but it cannot do so precisely due to our random method for inserting gadgets. The algorithm gets to decide how to split a gadget's probability of matching its innermost node towards a red edge or towards a blue edge, but after that the red edges get shuffled and the blue edges get shuffled and so any decision has a constant probability of getting messed up.

In order to have independence between gadget groups later, it will be convenient to run a Yao's minimax principle argument. Let Y be a random variable representing the algorithm's randomness (even shared public randomness between the nodes), and y be a specific value for this randomness. If the algorithm is deterministic, then this does not matter and we can say e.g. y can only be the empty string. If we condition on $Y = y$, then the algorithm behaves deterministically and in particular, every gadget's innermost node is deterministically matched to either a node in a red edge subgadget or a blue edge subgadget. After these decisions are locked in, we randomly insert all gadgets and consider what happens to our gadget groups.

We want to lower bound the probability that some node in the gadget group is unmatched. One sufficient condition for this to occur is that an even number of the five gadgets in the gadget group match towards the anchor node. The algorithm's decision (which can depend on Y) amounts to choosing which subset of gadgets get to participate, but participating gadgets have a probability of either 1/2 or 1/3 of changing the parity of the outcome. Since the probability of changing the parity is never greater than 1/2, this means that the even-parity probability after a coin flip (viewed as a weighted average) is closer to the even-parity probability before the flip than the odd-parity probability before the flip; hence the even-parity probability can never drop below 1/2. Furthermore, due to conditioning $Y = y$, all gadget groups are independent Bernoulli variables (possibly with different probabilities).

Against deterministic LOCAL algorithms the conditioning doesn't matter, and we choose a minimal $k = 4$ giving us a single gadget group. On average against our distribution, the algorithm leaves at least 1/2 of a node unmatched; we can begin to work out its multiplicative approximation ratio to be at least:

$$\begin{aligned}\frac{n}{n - 1/2} &= 1 + \frac{1}{2n - 1}\\ &> 1 + \frac{1}{2n}\\ &= 1 + \frac{1}{8k + 20k\Delta\rho}\\ &= 1 + \frac{1}{32 + 80\Delta\rho}\end{aligned}$$

To turn this into $(1 + \epsilon)$, we let $\rho = \lfloor \frac{\epsilon^{-1} - 32}{80\Delta} \rfloor$. This is $\Omega(\Delta^{-1}\epsilon^{-1})$ as claimed, and as long as $\epsilon < \frac{1}{160\Delta + 32}$ we have $\rho \geqslant 2$ so that our edge subgadgets are valid. Additionally our construction has $\Omega(\Delta^{-1}\epsilon^{-1})$ nodes which the algorithm was guaranteed to be able to handle and we guarded against 2ρ round algorithms which is $\Omega(\Delta^{-1}\epsilon^{-1})$ as well. The upshot is that after making these choices for k and ρ, we have contradicted the $(1 + \epsilon)$-multiplicative approximation guarantee of the deterministic LOCAL algorithm, completing that half of the proof.

We need to be a little careful rerunning our Chernoff bound, since the failure probability may not be exactly 1/2, although it turns out that a lower bound on the probability suffices. The expected number of gadget groups which fail is μ, which is between $\frac{1}{2}\frac{k}{4} = \frac{k}{8}$ and $\frac{k}{4}$ (the exact value in this range can depend on y). We need some multiplicative error to run our Chernoff bound argument, so let us consider the situation where at most $\frac{k}{16}$ fail. Let X be a random variable for the total number of gadget pair failures; since X is a sum of independent Bernoulli variables (independent conditioned on $Y = y$) we know that:

$$\begin{aligned}\mathbf{Pr}[X \leqslant (1-\gamma)\mu | Y = y] &\leqslant e^{-\gamma^2\mu/2} \qquad \forall \gamma \in (0,1)\\ \mathbf{Pr}[X \leqslant \mu/2 | Y = y] &\leqslant e^{-\mu/8}\\ \mathbf{Pr}[X \leqslant k/16 | Y = y] &\leqslant e^{-k/64}\end{aligned}$$

To arrive at a contradiction, we want to show that the left-hand side is strictly less than $1 - \delta$. Hence it suffices to show that:

$$\begin{aligned}e^{-k/64} &< 1 - \delta\\ -k/64 &< \ln(1-\delta)\\ k &> 64\ln(1-\delta)^{-1}\end{aligned}$$

We satisfy this by choosing $k = 4\lceil 16\ln(1-\delta)^{-1}\rceil + 4$; note that this is guaranteed to be a multiple of four and will contribute an asymptotic factor of $\Omega(\ln(1-\delta)^{-1})$. Since we have now forced $\mathbf{Pr}[X \leqslant k/16] < 1 - \delta$, we know that the complement probability is $\mathbf{Pr}[X > k/16] < \delta$. When this many failures occur, our Monte Carlo algorithm will have an approximation ratio which is strictly larger than:

$$\begin{aligned}\frac{n}{n - k/16} &= 1 + \frac{k/16}{n - k/16}\\ &= 1 + \frac{k/16}{4k + 10k\Delta\rho - k/16}\\ &= 1 + \frac{1}{64 + 160\Delta\rho - 1}\\ &> 1 + \frac{1}{64 + 160\Delta\rho}\end{aligned}$$

To turn this into $(1+\epsilon)$, we let $\rho = \lfloor \frac{\epsilon^{-1} - 64}{160\Delta} \rfloor$. This is $\Omega(\Delta^{-1}\epsilon^{-1})$ as claimed, and as long as $\epsilon < \frac{1}{320\Delta + 64}$, we have $\rho \geqslant 2$ so that our edge subgadgets are valid. Additionally our construction has $\Omega(\Delta^{-1}\epsilon^{-1}\ln(1-\delta)^{-1})$ nodes which the algorithm was guaranteed to be able to handle and we guarded against 2ρ round algorithms which is $\Omega(\Delta^{-1}\epsilon^{-1})$. Hence with probability strictly greater than δ (just over our counterexample distribution), conditioned on $Y = y$, the approximation ratio is strictly worse than $(1+\epsilon)$. But we now finish our Yao's minimax argument and observe that since this argument worked for every value of y, we can safely uncondition this statement by taking a weighted average. Hence with probability strictly greater than δ (over both our counterexample distribution and the randomness of the algorithm), the approximation ratio is strictly worse than $(1 + \epsilon)$.

This contradicts the approximation guarantee of the Monte Carlo LOCAL algorithm, completing the second part of the proof. $\square$

Chapter 6

Improved Inapproximability for MaxIS in CONGEST

In Chapter 4, we showed that an $O(\Delta)$-approximation to the Maximum Independent Set can be found in $O(\text{poly}(\log\log n))$ rounds in the CONGEST model. In this chapter, we demonstrate that achieving substantially better approximation factors, namely, $2-\epsilon$ and $4/3-\epsilon$, is impossible even in linear and quadratic times, respectively. This chapter is adapted from a joint work with Yuval Efron and Ofer Grossman [139].

6.1 Introduction

By far the most fruitful technique for showing lower bounds in the CONGEST model is reductions from two-party communication complexity, as discussed in Chapter 3. This technique has yielded nearly tight results for various fundamental problems, such as distance computations, minimum spanning tree, minimum vertex cover, and more [7,43,99,103,119,132,140,150,155,162,168,180,224,237,252].

In this work, we take this technique a step further, and we introduce a framework of reductions from t-party communication complexity, for every $t \geqslant 2$. Our framework enables us to show improved hardness of approximation results for Maximum Independent Set.

Bachrach et al. [43] used the two-party framework to show that finding a $(5/6+\epsilon)$-approximation requires $\Omega(n/\log^6 n)$ rounds, and finding a $(7/8+\epsilon)$-

approximation requires $\Omega(n^2/\log^7 n)$ rounds, in the CONGEST model. We amplify the hardness results of Bachrach et al. by using more parties. Our results:

Theorem 6.1.1. *For any constant $0 < \epsilon < 1/2$, any algorithm that finds a $(1/2+\epsilon)$-approximation for Maximum Independent Set in the CONGEST model requires at least $\Omega(n/\log^3 n)$ rounds.*

Theorem 6.1.2. *For any constant $0 < \epsilon < 1/4$, any algorithm that finds a $(3/4+\epsilon)$-approximation for Maximum Independent Set in the CONGEST model requires at least $\Omega(n^2/\log^3 n)$ rounds.*

While our results are not necessarily tight, we hope that our technique could pave the way for more and stronger lower bounds in the CONGEST model. We note our results hold even against randomized algorithms that succeed with probability $p \geqslant 2/3$, and even for constant diameter graphs. The hard instances that are used to prove Theorems 6.1.1 and 6.1.2 are weighted graphs, but we can extend our arguments for unweighted graphs as well by losing a logarithmic factor in the lower bounds (in terms of the number of rounds), as explained in Remark 6.4.9.

To prove Theorems 6.1.1 and 6.1.2 we use reductions from t-party communication complexity where we use $t = O(1/\epsilon)$ players. For $t = 2$, our constructions are similar to the ones presented in [43], and can be viewed as simplified versions of them.

The Challenge: Perhaps the first attempt that one would try to extend the two-party framework to the multi-party case is to use a reduction from the multi-party Set-Disjointness problem. In the multi-party Set-Disjointness problem, there are t players $p_1, \cdots, p_t$. Each player receives a string $x^i \in \{0,1\}^k$, and they wish to know if the strings all intersect at the same index. That is, they wish to know if there is an index $m \in [k]$ such that $x^1_m = x^2_m = \cdots = x^t_m = 1$. However, using a reduction from the multi-party Set-Disjointness problem is not a simple task, and as t increases, the task becomes more challenging. This complexity arises because, in the non-intersecting case, there are many sub-cases of pairwise intersections, and the reduction needs to account for all these sub-cases. For example, if we try to extend the reduction of [43] to the multi-party Set-Disjointness problem, in the non-intersecting case, for every pair $i \neq j \in [t]$, whether the strings x^i and x^j intersect influences the size (or weight) of the Maximum Independent Set. Therefore, in the non-intersecting case, the reduction needs to account for all the sub-cases of pairwise intersections. The more players we have, the more sub-cases arise, and the more infeasible the reduction becomes.

To overcome this challenge, we use reductions from a certain *Promise Pairwise Disjointness* problem, rather than the multi-party Set-Disjointness problem. In this Promise Pairwise Disjointness problem, there are t players each receiving a string $x^i \in \{0,1\}^k$, with the promise that the strings are either all intersecting at the same index or pairwise disjoint. That is, in the non-intersecting case, for all pairs $i \neq j \in [t]$, it holds that x^i and x^j are disjoint. Most importantly, we don't have many sub-cases of pairwise intersections in the non-intersecting case. The communication complexity of this Promise Pairwise Disjointness problem is $\Omega(k/t \log t)$ [105], which is large enough for our purposes, and we are able to use it to prove our results.

Road-map: In Section 6.2, we begin with some useful definitions and tools. In Section 6.3, we present our framework of reductions from the multi-party communication complexity model. The technical heart of this chapter is provided in Sections 6.4 and 6.5, where we show our linear and quadratic lower bounds, respectively.

6.2 Preliminaries

6.2.1 Multi-party Communication Complexity

Our lower bounds rely on reductions from the number-in-hand model of multi-party communication complexity. In the number-in-hand model, there are t players, each is holding an input $x^i \in \{0,1\}^k$, and they wish to compute a joint function of their inputs $f(x^1, \cdots, x^t)$, where t and k are parameters of the model. The communication setting in the number-in-hand model can be defined in various ways. In this work we use the shared blackboard model (see also, for example, [241]), where the players can exchange messages by writing them on a shared blackboard that is visible to all the players. The communication complexity in this model is formally defined as follows.

Definition 6.2.1. [Communication Complexity - Shared Blackboard]
Let $k \geqslant 1$, $t \geqslant 2$ be two integers, f be a Boolean function $f : \prod_{i=1}^{t}\{0,1\}^k \to \{\text{TRUE}, \text{FALSE}\}$, and $\mathcal{Q}$ be the family of protocols that compute f correctly with

probability at least 2/3, in the shared blackboard model. Given t inputs $x^1, \cdots, x^t$, denote by $\pi_Q(x^1, \cdots, x^t)$ the transcript of a protocol Q on the inputs $x^1, \cdots, x^t$, i.e. the sequence of bits that are written on the shared blackboard. The cost of a protocol Q is

$$Cost(Q) = \max_{x^1, \cdots, x^t \in \{0,1\}^k} |\pi_Q(x^1, \cdots, x^t)|$$

The communication complexity of f, denoted by $CC_f(k,t)$, is defined to be the minimum cost over all the possible protocols that compute f correctly with probability at least 2/3:

$$CC_f(k,t) = \min_{Q \in \mathcal{Q}} Cost(Q)$$

Our lower bounds for the CONGEST model are achieved via reductions from the *Promise Pairwise Disjointness* function. For two strings $x, y \in \{0,1\}^k$, we say that x and y are disjoint if $\sum_{j=1}^{k} x_j y_j = 0$.

Definition 6.2.2. [Promise Pairwise Disjointness]
Let $k \geqslant 1$, $t \geqslant 2$, and $x^1, \cdots, x^t \in \{0,1\}^k$, with the promise that the strings $x^1, \cdots, x^t$ are either *uniquely intersecting*, or pairwise disjoint. That is, either there is an $m \in [k]$ satisfying $x^1_m = x^2_m = \cdots = x^t_m = 1$, or x^i and x^j are disjoint for all pairs $i \neq j \in [t]$. The Promise Pairwise Disjointness function outputs TRUE if the strings are pairwise disjoint, and FALSE if they are uniquely intersecting.[1]

Chakrabarti et al. [105] proved the following theorem.

Theorem 6.2.3. *[Theorem 2.5 in [105]]*
Let f be the Promise Pairwise Disjointness function. It holds that $CC_f(k,t) = \Omega(k/t \log t)$.

6.2.2 Large Distance Codes

Our proofs use the tool of *error-correcting codes* that was used in [43]. Let us define the notion of a *code-mapping*. Here, we use a similar definition to the one given by Arora and Barak [34] (Chapter 19, Definition 19.5, page 380, in [34]).

[1]For any positive integer k, we denote by $[k]$ the set of positive integers $\{1, 2, \cdots, k\}$.

Definition 6.2.4. [Code-mapping]
Let Σ be a finite set of symbols, called the alphabet. Fix three integers $d \geqslant 1$, $L \geqslant 1$ and $M \geqslant L$. For two strings $x, y \in \Sigma^M$, the distance of x and y, denoted by $d(x,y)$, is equal to $|\{i \in [M] \mid x_i \neq y_i\}|$.

A code-mapping with parameters (L, M, d, Σ) is a function $\mathcal{C} : \Sigma^L \to \Sigma^M$, such that for every $x \neq y \in \Sigma^L$, $d(\mathcal{C}(x), \mathcal{C}(y)) \geqslant d$.

Our proofs use the following Theorem that shows the existence of large-distance codes (Lemma 19.11 in [34]).

Theorem 6.2.5. *Let Σ be an alphabet of size $q = |\Sigma|$. There is a code-mapping with parameters (L, M, d, Σ), where $L \leqslant M \leqslant q$ and $d = M - L$.*

One way to construct a code-mapping that proves Theorem 6.2.5 is by the so called *Reed-Solomon* code, which is a well-known algebraic construction for error-correcting codes. In our proofs we don't need the details of the construction, but only its existence.

6.3 Multi-Party Communication Complexity Reductions

In this section we show how to prove lower bounds for the CONGEST model via reductions from the shared blackboard model of multi-party communication complexity. Our framework extends the notion of a family of lower bound graphs [103] that was discussed in Chapter 3 (in Definition 3.2.4), for any arbitrary number $t \geqslant 2$ of players.

Definition 6.3.1. [Family of Lower Bound Graphs]
Given two integers $k \geqslant 1$, $t \geqslant 2$, a function $f : \prod_{i=1}^{t} \{0,1\}^k \to \{\text{TRUE}, \text{FALSE}\}$, and a graph predicate P, a family of graphs

$$\{ G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} = (x^1, \cdots, x^t) \in \prod_{i=1}^{t} \{0,1\}^k \}$$

is said to be a family of *lower bound graphs with respect to f and P* if there is a partition of the set of nodes $V = \dot{\bigcup}_{i=1}^{t} V^i$ for which the following properties hold:[2]

[2]Throughout this chapter, we use the notation $V = \dot{\bigcup}_{i=1}^{t} V^i$ to emphasize that $\{V^i\}_{i \in [t]}$ is a partition of V.

1. Only the weight of the nodes in V^i and the existence of edges in $V^i \times V^i$ may depend on x^i;

2. $G_{\bar{x}}$ satisfies the predicate P iff $f(\bar{x}) = \text{TRUE}$.

Next, we prove the following reduction theorem, which is based on a standard simulation argument. This theorem extends Theorem 3.2.5 that was stated in Chapter 3 (and also in Theorem 1 in [103]). Given a family of lower bound graphs and a graph $G_{\bar{x}}$ in it, we denote by $cut(G_{\bar{x}})$ the set of *cut edges* of $G_{\bar{x}}$. That is, $cut(G_{\bar{x}}) = E_{\bar{x}} \setminus (\bigcup_{i=1}^{t} V^i \times V^i)$.

Theorem 6.3.2. *Fix $k \geqslant 1$, $t \geqslant 2$, $f : \prod_{i=1}^{t}\{0,1\}^k \to \{\text{TRUE}, \text{FALSE}\}$, and a graph predicate P. If there is a family $\{G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}})\}$ of lower bound graphs w.r.t. f and P, then any algorithm for deciding P in the CONGEST model with success probability at least $2/3$ requires $\Omega\left(\frac{CC_f(k,t)}{|cut(G_{\bar{x}})|\log|V|}\right)$ rounds.*

Proof. Let ALG be a distributed algorithm in the CONGEST model that decides P in T rounds. We define a protocol for f in the shared blackboard model, as follows. Let $\bar{x} = (x^1, \cdots, x^t) \in \prod_{i=1}^{t}\{0,1\}^k$ be the vector of inputs of the players $p^1, \cdots, p^t$, where p^i receives the string x^i, in the shared blackboard model. Each player p^i constructs the part of $G_{\bar{x}}$ for the nodes in V^i. This can be done by the first condition of Definition 6.3.1, and the fact that the V^i's are disjoint.

The players $p^1, \cdots, p^t$ simulate ALG, where each player p^i simulates the nodes in V^i, as follows. All the messages that are sent on edges in $V^i \times V^i$ are simulated by player p^i, without any communication with the other players. All the other messages, the ones that are sent on edges in $cut(G_{\bar{x}}) = E_{\bar{x}} \setminus \left(\bigcup_{i=1}^{t} V^i \times V^i\right)$, are written on the shared blackboard. That is, whenever there is a message from some node in V^i to some node in V^j for $i \neq j \in [t]$, player p^i writes this message on the shared blackboard, which is visible to all the other players. In particular, it is visible to p^j who is simulating the nodes in V^j.

After simulating the T rounds of ALG, the players know whether $G_{\bar{x}}$ satisfies the predicate P, and by the second condition of Definition 6.3.1, this reveals the information about $f(\bar{x})$. Observe that the total number of bits that are written on the blackboard are $O(T|cut(G_{\bar{x}})|\log|V|)$. This is because an algorithm in the CONGEST model sends at most $O(\log|V|)$ bits on each edge in each round, and the only messages that are written on the blackboard are the ones that are sent on the edges in $cut(G_{\bar{x}})$. Hence, the communication complexity of f is at most $O(T|cut(G_{\bar{x}})|\log|V|)$ and therefore, $T = \Omega\left(\frac{CC_f(k,t)}{|cut(G_{\bar{x}})|\log|V|}\right)$. □

Our hardness results use families of lower bound graphs with respect to the Promise Pairwise Disjointness function and a gap predicate P. We formalize such families in Definition 6.3.4. First, we formally define the notion of γ-approximation for Maximum Independent Set.

Definition 6.3.3. [γ-approximation for Maximum Independent Set]
Let $G = (V, E, w)$ be a vertex-weighted graph with weight function w, and let OPT be the value of an optimal solution for Maximum Independent Set.[3] An independent set I in G is γ-approximation for Maximum Independent Set if $w(I) \geqslant OPT/\gamma$.

Definition 6.3.4. [γ-approximate MaxIS family of lower bound graphs]
Fix $0 \leqslant \gamma \leqslant 1, \beta > 0$. Let P be a graph predicate that distinguishes between graphs of Maximum Independent Set of weight at least β, and graphs of Maximum Independent Set of weight at most $\gamma \cdot \beta$. A family of graphs is called a γ-approximate MaxIS if it is a family of lower bound graphs with respect to the Promise Pairwise Disjointness function and the graph predicate P.

The following corollary follows from Theorems 6.2.3 and 6.3.2.

Corollary 6.3.5. *Let $k \geqslant 1$, $t \geqslant 2$ be two integers. If there is a γ-approximate MaxIS family of graphs $\{G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t}\{0,1\}^k\}$, then any algorithm for finding a γ-approximation of Maximum Independent Set in the CONGEST model with success probability at least $2/3$ requires $\Omega(k/(t\log t \cdot |cut(G_{\bar{x}})|\log|V|))$ rounds.*

6.4 Linear Lower Bound

In this section we prove the following theorem.

Theorem 6.1.1 *For any constant $0 < \epsilon < 1/2$, any algorithm that finds a $(1/2+\epsilon)$-approximation for Maximum Independent Set in the CONGEST model requires at least $\Omega(n/\log^3 n)$ rounds.*

[3]Throughout this chapter, for a subset of nodes $U \subseteq V$, we denote by $w(U) = \sum_{v \in U} w(v)$.

In order to prove Theorem 6.1.1, we construct a $(1/2+\epsilon)$-approximate *MaxIS* family of lower bound graphs $\{ G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t}\{0,1\}^k \}$.

6.4.1 The family of lower bound graphs

We start by describing a fixed graph construction $G = (V, E, w)$, and then we describe how to get from G and a vector of strings $\bar{x} \in \prod_{i=1}^{t}\{0,1\}^k$ the graph $G_{\bar{x}} = (V, E_{\bar{x}}, x_{\bar{x}})$, which gives a family of graphs $\{ G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t}\{0,1\}^k \}$.

Our fixed graph construction G contains t copies of a fixed *base graph* H. We start by describing the base graph H.

Some notations. Let k, α, ℓ be three positive integers that are to be chosen later such that $(\ell+\alpha)^\alpha = k$, and $\ell \gg \alpha$. Let $\mathcal{C}$ be a code-mapping given by Theorem 6.2.5 with parameters $(\alpha, \ell+\alpha, \ell, \Sigma)$, where $\Sigma = \{1, \cdots, \ell+\alpha\}$. Observe that $k = |\Sigma|^\alpha$. Hence, we order the elements in Σ^α by an arbitrary ordering, and for $m \in [k]$, we denote by $\mathcal{C}(m)$ the code-mapping of the m'th element in Σ^α.

Description of $H = (V_H, E_H)$. The set of nodes V_H contains a clique of size k, denoted by $A = \{v_1, ..., v_k\}$, and $\ell+\alpha$ cliques, $C_1, \cdots, C_{(\ell+\alpha)}$, each of size $\ell+\alpha$. For each $h \in [\ell+\alpha]$, the nodes in C_h are denoted by $C_h = \{\sigma_{(h,1)}, \cdots, \sigma_{(h,\ell+\alpha)}\}$. We call the cliques $C_1, \cdots, C_{\ell+\alpha}$ the *code gadget*, and we denote this set of nodes by

$$Code = \bigcup_{h=1}^{\ell+\alpha} C_h$$

The reason that these cliques are called the code-gadget is as follows. Given a code-word $w \in \Sigma^{\ell+\alpha}$, we can represent w by $\ell+\alpha$ nodes

$$u_1 \in C_1, u_2 \in C_2, \cdots, u_{\ell+\alpha} \in C_{\ell+\alpha}$$

where $u_h \in C_h$ corresponds to the h'th position in w. That is, $u_h = \sigma_{(h,w_h)}$, where w_h is the value in the h'th position in w. For any $m \in [k]$, we denote by $Code_m$ the set of nodes that corresponds to the code-word $\mathbb{C}(m) \in \Sigma^{\ell+\alpha}$, and we connect $v_m \in A$ to all the nodes in $Code \setminus Code_m$.

This concludes the description of H. More formally, the graph $H = (V_H, E_H)$ is defined as follows. Given a clique C, we denote by $E(C)$ the set of all the possible

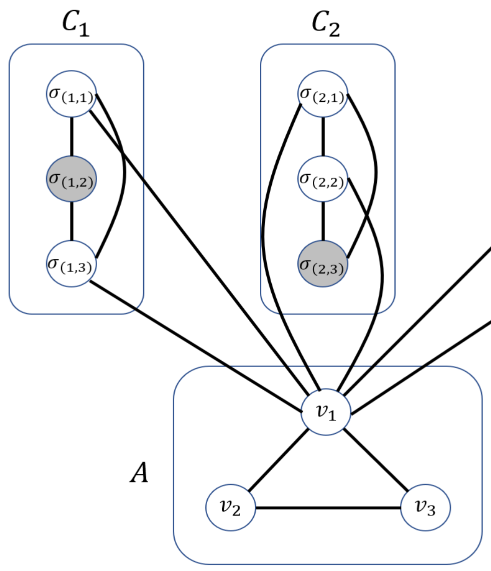

Figure 6.1: An example of the base graph H, where $\ell = 2$, $\alpha = 1$. A is a clique of $k = (\ell + \alpha)^\alpha = 3$ nodes, and there are $\ell + \alpha = 3$ cliques C_1, C_2, and C_3, each of size 3. In this example, we assume that the code-mapping of 1, $\mathbb{C}(1) =$ "2, 3, 1", and therefore, v_1 in connected to all the nodes in $Code = C_1 \cup C_2 \cup C_3$, except of the nodes in $Code_1 = \{\sigma_{(1,2)}, \sigma_{(2,3)}, \sigma_{(3,1)}\}$. The other edges between $\{v_2, v_3\} \times Code$ are omitted in this figure, for clarity.

edges between nodes in C.

$$V_H = A \cup Code$$

$$E_H = E(A) \cup \{\{v_m, u\} \mid v_m \in A, u \in Code \setminus Code_m\} \bigcup_{h=1}^{\ell+\alpha} E(C_h)$$

Obtaining the fixed graph construction G from H: Now we are ready to describe the fixed graph construction $G = (V, E)$. Let $t \geqslant 2$. There are t copies of H in G, denoted by $H^1, \cdots, H^t$. In order to distinguish between nodes in different H^i's, we add a superscript i for the nodes in H^i. That is, for each $i \in [t]$, $H^i = (V^i, E^i)$ contains a clique and a code-gadget, where the clique is denoted by $A^i = \{v^i_1, \cdots, v^i_k\}$, the code-gadget is denoted by $Code^i$, the cliques in the code-gadget are denoted by $C^i_1, \cdots, C^i_{\ell+\alpha}$, and for any $h \in [\ell + \alpha]$, the nodes in C^i_h are denoted by $C^i_h = \{\sigma^i_{(h,1)}, \cdots, \sigma^i_{(h,\ell+\alpha)}\}$. Similarly, $Code^i_m$ denotes the set of nodes

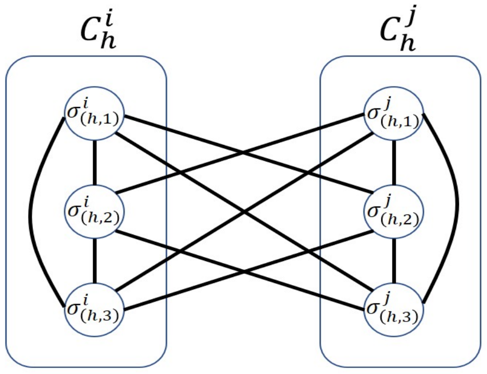

Figure 6.2: An illustration for the connections between C_h^i and C_h^j. In this example, $\ell + \alpha = 3$. Observe that for any $r \in \{1, 2, 3\}$, $\sigma^i_{h,r}$ is connected to all the nodes in C_h^j except of $\sigma^j_{h,r}$.

in $\bigcup_{i=1}^{\ell+\alpha} C_h^i$ that corresponds to the code-word $\mathcal{C}_m$. That is, let $w = \mathbb{C}(m)$, we have that $Code^i_m = \{\sigma^i_{(1,w_1)}, \cdots, \sigma^i_{(\ell+\alpha, w_{\ell+\alpha})}\}$.

It remains to describe the connections between H^i and H^j, for any $i \neq j \in [t]$. For any $h \in [\ell + \alpha]$, we add all the possible edges between C_h^i and C_h^j *except of the natural perfect matching between* C_h^i *and* C_h^j, i.e., $\{\{\sigma^i_{(h,r)}, \sigma^j_{(h,r)}\} \mid r \in [\ell + \alpha]\}$. More formally, we add the following edges for any $i \neq j \in [t]$ and any $h \in [\ell + \alpha]$,

$$\left\{\{u, v\} \mid u \in C_h^i, v \in C_h^j\right\} \setminus \left\{\{\sigma^i_{(h,r)}, \sigma^j_{(h,r)}\} \mid r \in [\ell + \alpha]\right\}$$

This concludes our fixed graph construction G, and we proceed to describing $G_{\bar{x}}$.

Obtaining $G_{\bar{x}}$ from G and $\bar{x}$: Given $\bar{x} = (x^1, \cdots, x^t) \in \prod_{i=1}^t \{0,1\}^k$. The graph $G_{\bar{x}} = (V, E, w_{\bar{x}})$ is defined as follows. The sets of nodes and edges of $G_{\bar{x}}$ are exactly as in G. The weights of nodes in $G_{\bar{x}}$ are defined as follows. Let $i \in [t]$, $m \in [k]$, and $v^i_m \in A^i$,

$$w(v^i_m) = \begin{cases} \ell & \text{if } x^i_m = 1 \\ 1 & \text{otherwise} \end{cases}$$

All the other nodes in $G_{\bar{x}}$ are of weight 1. That is, for any $u \in V \setminus \bigcup_{i=1}^{t} A^i$, $w(u) = 1$.

This concludes the description of $G_{\bar{x}}$. Before we proceed to proving that $G_{\bar{x}}$ is a family of lower bound graphs, we provide three useful properties of $G_{\bar{x}}$ that are used in the proof.

Property 3. *For any $m \in [k]$, it holds that $\left(\bigcup_{i=1}^{t} Code_m^i\right) \cup \{v_m^i \mid i \in [t]\}$ is an independent set.*

Proof. First, observe that the nodes in $\{v_m^i \mid i \in [t]\}$ are independent. This is because $v_m^i \in A^i$, and there are no edges between A^i and A^j for any $i \neq j$. There are also no edges between A^i and $Code^j$, for any $i \neq j$. Furthermore, for any $i \in [t]$ and any $m \in [k]$, it holds that $\{v_m^i\} \cup Code_m^i$ is an independent set. This is because v_m^i is connected only to the nodes in $Code^i \setminus Code_m^i$. Finally, let $w = \mathbb{C}(m)$ be the code-mapping of m. Since for any $i \neq j$, we have that $Code_m^i = \{\sigma^i_{(1,w_1)}, \cdots \sigma^i_{(\ell+\alpha, w_{\ell+\alpha})}\}$, and $Code_m^j = \{\sigma^j_{(1,w_1)}, \cdots \sigma^j_{(\ell+\alpha, w_{\ell+\alpha})}\}$, and $\sigma^i_{(h,r)}$ is not connected to $\sigma^j_{(h,r)}$ for any $h, r \in [\ell + \alpha]$, we have that $\bigcup_{i=1}^{t} Code_m^i$ is an independent set. Hence, the union $\left(\bigcup_{i=1}^{t} Code_m^i\right) \cup \{v_m^i \mid i \in [t]\}$ is an independent set. □

Property 4. *For any $i \neq j \in [t]$, and any $m_1 \neq m_2 \in [k]$, the bipartite graph $(Code_{m_1}^i, Code_{m_2}^j)$ contains a matching of size at least ℓ.*

Proof. Let $w^1 = \mathbb{C}(m_1)$ be the code-mapping of m_1, and let the $w^2 = \mathbb{C}(m_2)$ be the code-mapping of m_2. Given $h, r \in [\ell + \alpha]$, observe that $\sigma^i_{(h,r)}$ is connected to all the nodes in $C_h^j \setminus \{\sigma^j_{(h,r)}\}$. Hence, since the distance between w^1 and w^2 is at least ℓ, there are at least ℓ positions $h \in [\ell + \alpha]$ for which $w_h^1 \neq w_h^2$, and therefore, there are at least ℓ positions $h \in [\ell + \alpha]$ for which it holds that $\sigma^i_{(h,w_h^1)}$ is connected to $\sigma^j_{(h,w_h^2)}$, where w_h^1 is the h'th position in w^1 and w_h^2 is the h'th position in w^2. □

Property 5. *Let $i \neq j \in [t]$, let $m_1 \neq m_2 \in [k]$, and let I be any independent set. Let $w^1 = \mathbb{C}(m_1)$ be the code mapping of m_1, and let $w^2 = \mathbb{C}(m_2)$ be the code-mapping of m_2. The number of positions $h \in [\ell + \alpha]$ for which it holds that $\sigma^i_{(h,w_h^1)} \in I$ and $\sigma^j_{(h,w_h^2)} \in I$ is at most α.*

Proof. By Property 4, the bipartite graph $(Code_{m_1}^i, Code_{m_2}^j)$ contains a matching of size at least ℓ. Therefore, there are at least ℓ positions $h \in [\ell + \alpha]$ for which I

contains at most one of the nodes $\sigma^i_{(h,w^1_h)}$ and $\sigma^j_{(h,w^2_h)}$. This leaves at most α other positions for which I can contain both $\sigma^i_{(h,w^1_h)}$ and $\sigma^j_{(h,w^2_h)}$. □

6.4.2 $G_{\bar{x}}$ is a $(1/2+\epsilon)$-approximate $MaxIS$ family of lower bound graphs

In this section we show that there is a constant $t > 2$ for which $G_{\bar{x}}$ is a $(1/2+\epsilon)$-approximate $MaxIS$ family of graphs. We start with a slightly weaker statement for $t = 2$, which is later used in the proof for $t > 2$.

6.4.2.1 Warm-up: $t = 2$

In this section we prove the following lemma.

Lemma 6.4.1. *For $t = 2$, and for any constant $\epsilon > 0$, it holds that the family of graphs*

$$\{\, G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t} \{0,1\}^k \,\}$$

is a $(3/4+\epsilon)$-approximate $MaxIS$ family of lower bound graphs.

For the rest of this subsection, we assume that $t = 2$. Lemma 6.4.1 is a corollary of Claims 6.4.2 and 6.4.3.

Claim 6.4.2.
For any $g_{(x^1,x^2)} \in \{\, G_{(x^1,x^2)} = (V, E_{(x^1,x^2)}, w_{(x^1,x^2)}) \mid (x^1,x^2) \in \prod_{i=1}^{2}\{0,1\}^k \,\}$, if x^1 and x^2 are not disjoint, then $g_{(x^1,x^2)}$ contains an independent set of weight at least $4\ell + 2\alpha$.

Proof. Since the sets are not disjoint, there is an $m \in [k]$ for which $x^1_m = x^2_m = 1$. Therefore, the weight of each of the nodes v^1_m and v^2_m is ℓ. By Property 3, the set $\{v^1_m\} \cup \{v^2_m\} \cup Code^1_m \cup Code^2_m$ is independent, and observe that its weight is $4\ell + 2\alpha$. □

Claim 6.4.3.
For any $g_{(x^1,x^2)} \in \{\, G_{(x^1,x^2)} = (V, E_{(x^1,x^2)}, w_{(x^1,x^2)}) \mid (x^1,x^2) \in \prod_{i=1}^{2}\{0,1\}^k \,\}$, if x^1 and x^2 are disjoint, then any independent set I in $g_{(x^1,x^2)}$ is of weight at most $3\ell + 2\alpha + 1$.

Proof. The proof is by the following simple case analysis.

1. I contains at most one node of weight ℓ: In this case, the node of weight ℓ must be either in the clique A^1 or in the clique A^2. Assume without loss of generality that this node is in A^1. Observe that we can take at most one node of weight 1 from A^2. Furthermore, since each of $Code^1$ and $Code^2$ is a union of $\ell + \alpha$ cliques, we cannot construct an independent set in $Code^1 \cup Code^2$ of weight larger than $2(\ell + \alpha)$, it follows that the weight of I cannot be larger than $3\ell + 2\alpha + 1$.

2. I contains two nodes of weight ℓ: This implies that I contains one node $v^1_{m_1} \in A^1$ of weight ℓ and another node $v^2_{m_2} \in A^2$ of weight ℓ, where $m_1 \neq m_2 \in [k]$. Since the strings x^1, x^2 are disjoint, it must be the case that $m_1 \neq m_2$. Furthermore, since $v^1_{m_1}$ is connected to the nodes in $Code^1 \setminus Code^1_{m_1}$, and $v^2_{m_2}$ is connected to the nodes in $Code^2 \setminus Code^2_{m_2}$, it remains to show that $|I \cap (Code^1_{m_1} \cup Code^2_{m_2})| \leqslant \ell + 2\alpha$. By Property 4, $(Code^1_{m_1}, Code^2_{m_2})$ contains a matching of size at least ℓ, and since $|Code^1_{m_1} \cup Code^2_{m_2}| = (2\ell + 2\alpha)$, this implies that $|I \cap (Code^1_{m_1} \cup Code^2_{m_2})| \leqslant \ell + 2\alpha$. To conclude, in this case, I contains 2 nodes of weight ℓ and $\ell + 2\alpha$ nodes of weight 1. In total, the weight of I is $3\ell + 2\alpha$.

Notice that I cannot contain more than 2 elements of weight ℓ since the elements of weight ℓ form two disjoint cliques. □

Proof of Lemma 6.4.1. Claims 6.4.2 and 6.4.3 imply that the family of graphs $\{ G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{2}\{0,1\}^k \}$ is a family of lower bound graphs with respect to the set disjointness function and the graph predicate that distinguishes between graphs of Maximum Independent Set at least $4\ell + 2\alpha$ and graphs of Maximum Independent Set at most $3\ell + 2\alpha + 1$.

We set $\ell = \log k - \log k / \log\log k, \alpha = \log k / \log\log k$. Hence $(\ell + \alpha)^{\alpha} = k$ as desired. Since the dominating terms in the two cases are 4ℓ and 3ℓ, it follows that for any constant $\epsilon > 0$, $\{ G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{2}\{0,1\}^k \}$ is a $(3/4 + \epsilon)$-approximate *MaxIS* family of graphs[4]. □

6.4.2.2 Hardness Amplification using $t > 2$ Players

In this section we prove the following lemma.

[4] In fact, by slightly changing the parameters ℓ and α, the claim holds for any $\epsilon = \Omega(1/\log k)$.

Lemma 6.4.4. *For any constant $\epsilon > 0$, there is a constant $t > 2$ for which it holds that $\{ G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t} \{0,1\}^k \}$ is a $(1/2+\epsilon)$-approximate MaxIS family of lower bound graphs.*

Lemma 6.4.4 follows from Claims 6.4.5 and 6.4.8.

Claim 6.4.5. For $t \geqslant 0$, and any $g_{\bar{x}} \in \{ G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t} \{0,1\}^k \}$, if there is an $m \in [k]$ for which it holds that $x_m^1 = \cdots = x_m^t = 1$, then $g_{\bar{x}}$ contains an independent set of weight at least $t(2\ell + \alpha)$.

Proof. Observe that for any $i \in [t]$, it holds that $w(v_m^i) = \ell$. Furthermore, by Property 3, $\left(\bigcup_{i=1}^{t} Code_m^i\right) \cup \{v_m^i \mid i \in [t]\}$ is an independent set, and it is of weight $2t\ell + t\alpha$. □

Before we proceed to the case in which the strings are pairwise disjoint, let us prove the following helper claim and a corollary of it.

Claim 6.4.6. For any positive integer t. Let $m_1, m_2, \cdots, m_t$ be any t distinct values in $[k]$. For any independent set I, if $\{v_{m_i}^i \mid i \in [t]\} \subseteq I$, then

$$\left| I \cap \left(\bigcup_{i=1}^{t} Code_{m_i}^i \right) \right| \leqslant \ell + \alpha t^2$$

Proof. Let us start with some notations. Let $w^i = \mathbb{C}(m_i)$ be the code-mapping of m_i. Hence, we have that $Code_{m_i}^i = \{\sigma^i_{(1,w_1^i)}, \cdots, \sigma^i_{(\ell+\alpha, w^i_{\ell+\alpha})}\}$. Furthermore, let $S = \{h \in [\ell+\alpha] \mid \sum_{i=1}^{t} |I \cap \sigma^i_{(h,w_h^i)}| \leqslant 1\}$, $\bar{S} = [\ell+\alpha] \setminus S$. That is, S is the set of values $h \in [\ell+\alpha]$ for which the independent set I contains at most one node in $\bigcup_{i=1}^{t} \{\sigma^i_{(h,w_h^i)}\}$. Finally, let $\psi_{i,j}^h$ be an indicator defined as follows.

$$\psi_{i,j}^h = \begin{cases} 1 & \text{if } \sigma^i_{(h,w_h^i)} \in I \text{ and } \sigma^j_{(h,w_h^j)} \in I \\ 0 & \text{otherwise} \end{cases}$$

By Property 5, for any $i \neq j \in [t]$, it holds that $\sum_{h \in [\alpha+\ell]} \psi_{i,j}^h \leqslant \alpha$. Hence,

$$\left| I \cap \left(\bigcup_{i=1}^{t} Code_{m_i}^i \right) \right| = \sum_{i=1}^{t} \left| I \cap Code_{m_i}^i \right| = \sum_{i=1}^{t} \left| I \cap \left(\bigcup_{h=1}^{\ell+\alpha} \{\sigma^i_{(h,w_h^i)}\} \right) \right| \tag{1}$$

$$
\begin{aligned}
&= \sum_{i=1}^{t}\sum_{h=1}^{\ell+\alpha} \left|I \cap \{\sigma^i_{(h,w^i_h)}\}\right| = \sum_{h=1}^{\ell+\alpha}\sum_{i=1}^{t} \left|I \cap \{\sigma^i_{(h,w^i_h)}\}\right| && (2)\\
&= \left(\sum_{h\in S}\sum_{i=1}^{t} \left|I \cap \{\sigma^i_{(h,w^i_h)}\}\right|\right) + \left(\sum_{h\in \bar{S}}\sum_{i=1}^{t} \left|I \cap \{\sigma^i_{(h,w^i_h)}\}\right|\right) && (3)\\
&\leqslant \left(\sum_{h\in S}\sum_{i=1}^{t} \left|I \cap \{\sigma^i_{(h,w^i_h)}\}\right|\right) + \left(\sum_{h\in \bar{S}}\sum_{i\neq j\in[t]} 2\psi^h_{i,j}\right) && (4)\\
&= \left(\sum_{h\in S}\sum_{i=1}^{t} \left|I \cap \{\sigma^i_{(h,w^i_h)}\}\right|\right) + \left(\sum_{i\neq j\in[t]}\sum_{h\in \bar{S}} 2\psi^h_{i,j}\right) && (5)\\
&\leqslant \ell + \alpha + 2\alpha \cdot t(t-1)/2 \leqslant \ell + \alpha t^2 && (6)
\end{aligned}
$$

Where (1), (2), and (3) are straightforward. (4) holds because for any $h \in \bar{S}$, there are at least two indices $i \neq j \in [t]$, for which it holds that $\sigma^i_{(h,w^i_h)} \in I$ and $\sigma^j_{(h,w^j_h)} \in I$. (5) holds by changing the summation order of the second sum. (6) holds because for any $h \in S$, $\sum_{i=1}^{t} |I \cap \{\sigma^i_{(h,w^i_h)}\}| \leqslant 1$, and by Property 3, $\sum_{h\in[\ell+\alpha]} \psi^h_{i,j} \leqslant \alpha$. $\square$

Corollary 6.4.7. *For any positive integer t, let $m_1, m_2, \cdots, m_t$ be any t distinct values in $[k]$. For any independent set I, if $\{v^i_{m_i} \mid i \in [t]\} \subseteq I$, then*

$$w(I) \leqslant (t+1)\ell + \alpha t^2$$

Proof. Since each $v^i_{m_i}$ is connected to all the nodes in $Code^i \setminus Code^i_{m_i}$, we have that

$$w(I) = w(I \cap (\bigcup_{i=1}^{t} A_i)) + w(I \cap (\bigcup_{i=1}^{t} Code^i)) = \left(\sum_{i=1}^{t} w(v^i_{m_i})\right) + w(I \cap (\bigcup_{i=1}^{t} Code^i_{m_i}))$$
$$\leqslant t\ell + \ell + \alpha t^2 = (t+1)\ell + \alpha t^2$$

$\square$

Claim 6.4.8. For $t \geqslant 0$, and any $g_{\bar{x}} \in \{\, G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t}\{0,1\}^k \,\}$, if the strings $x^1, \cdots, x^t$ are pairwise disjoint, then the weight of any independent set is at most $(t+1)\ell + \alpha t^2$.

Proof. The proof is by induction on t, where the base case of $t = 1$ is straightforward (even the case of $t = 2$ was already proved in Claim 6.4.3). We assume correctness for $t-1$, and prove correctness for t. Let I be an independent set in $g_{\bar{x}}$. Recall that A^i is a clique and therefore $|I \cap A^i| \leqslant 1$. The proof is by the following case analysis.

1. There is some $i \in [t]$ for which it holds that $I \cap A^i$ is either empty, or contains a node of weight 1: Observe that in this case, $w(I \cap V^i) \leqslant \ell + \alpha + 1$. This is because any independent set contains at most $\ell + \alpha$ nodes in $Code^i = V^i \setminus A^i$. Furthermore, by the inductive hypothesis on the graph induced by the nodes in $\bigcup_{j \in [t] \setminus \{i\}} V^j$, we have that $w(I) \leqslant t\ell + \alpha(t-1)^2 + \ell + \alpha + 1 \leqslant (t+1)\ell + \alpha(t^2 - 2t + 1) + \alpha + 1 < (t+1)\ell + \alpha(t^2)$. Where the last inequality holds since $\alpha \geqslant 1$, and $t > 2$.

2. For any $i \in [t]$, $I \cap A^i$ contains a node of weight ℓ, denoted by $v^i_{m_i}$: This case is proved directly, without applying the inductive hypothesis, as follows. First, since the strings $x^1, \cdots, x^t$ are pairwise disjoint, it must be the case that for any $i \neq j \in [t]$, $m_i \neq m_j$. This is because $w(v^i_{m_i}) = \ell$ if and only if $x^i_{m_i} = 1$, and if $m_i = m_j$, it would imply that x^i and x^j are not disjoint. Hence, by Corollary 6.4.7, we have that

$$w(I) \leqslant (t+1)\ell + \alpha t^2$$

As desired.

□

Proof of Lemma 6.4.4. By Claims 6.4.5 and 6.4.8, we have that the family of graphs $\{ G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t} \{0,1\}^k \}$ is a family of lower bound graphs with respect to the pairwise disjointness function and the graph predicate that distinguishes between graphs of Maximum Independent Set at least $t(2\ell + \alpha)$ and graphs of Maximum Independent Set at most $(t+1)\ell + \alpha \cdot t^2$.

Recall that $\ell = \log k - \log k / \log\log k$, $\alpha = \log k / \log\log k$. Which implies that the graph predicate distinguishes between independent sets of weight at least $2t(\log k - \log k / \log\log k + \log k / \log\log k) = 2t \log k$ and independent sets of weight at most $(t+1)(\log k - \log k / \log\log k) + t^2(\log k / \log\log k) \leqslant (t+2)\log k$, for any constant t and $k \gg t$. Hence, for any constant $\epsilon > 0$, we choose $t = 2/\epsilon$ (or the first integer larger than $2/\epsilon$, if it is not an integer). This implies that for any constant $\epsilon > 0$, there is a constant t for which $\{ G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t} \{0,1\}^k \}$ is a $(1/2 + \epsilon)$-approximate *MaxIS* family of graphs. □

Proof of Theorem 6.1.1. Observe that $k = \Theta(n)$, where $n = |V|$. Furthermore, $\{ G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t} \{0,1\}^k \}$ is a $(1/2+\epsilon)$-approximate $MaxIS$ family of graphs, where the partition of the set of nodes that is needed for Definition 6.3.1 is $V = \bigcup_{i=1}^{t} V^i$. Hence, by Corollary 6.3.5 and the fact that $|cut(G_{\bar{x}})| = t^2 \log^2 k = \Theta(\log^2 k)$, any algorithm for finding a $(1/2+\epsilon)$-approximation for Maximum Independent Set in the CONGEST model with success probability at least 2/3 requires $\Omega(k/(t \log t \cdot |cut(G_{\bar{x}})| \log |V|)) = \Omega(n/(t \log t \cdot \log^3 n) = \Omega(n/\log^3 n)$ rounds. □

Remark 6.4.9. While our hard instances in the proof of Theorem 6.1.1 are weighted, it is easy to extend the argument for unweighted graphs as well, by losing a logarithmic factor in the lower bound (in terms of the number of rounds), as follows. For every node v of weight ℓ, we replace v by an independent set of size ℓ, denoted by $I(v)$. For every node u that is adjacent to v in our construction, if u is of weight 1, we connect all the nodes in $I(v)$ to u. Otherwise, if u is of weight ℓ, it means that it is replaced by an independent set of size ℓ, denoted by $I(u)$. We connect $I(v)$ to $I(u)$ by a bi-clique (a full bipartite graph). The proof that the converted construction yields a hardness of $(1/2+\epsilon)$-approximation follows from a similar case analysis to the one provided for the weighted case. Since the number of nodes in the unweighted construction in $n = \Theta(k\ell) = \Theta(k \log k)$ rather than $\Theta(k)$, in terms of the number of rounds, we lose a logarithmic factor in the lower bound compared to the weighted case.

6.5 Quadratic Lower Bound

In this section we prove the following theorem.

Theorem 6.1.2 *For any constant $0 < \epsilon < 1/4$, any algorithm that finds a $(3/4+\epsilon)$-approximation for Maximum Independent Set in the CONGEST model requires at least $\Omega(n^2/\log^3 n)$ rounds.*

In order to prove Theorem 6.1.2, we construct a $(3/4+\epsilon)$-approximate $MaxIS$ family of lower bound graphs $\{ F_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t} \{0,1\}^{k^2} \}$. Observe that unlike the previous section, the length of the strings in $\bar{x}$ is k^2 rather than k. In our graph construction, similarly to the previous section, $k = \Theta(n)$. Hence, having the length of the strings being k^2 allows us to achieve a near-quadratic lower bound. Our hard instances are weighted graphs, and we can extend our argument to unweighted graphs as well by losing a logarithmic factor in the lower bound (in terms of the number of rounds) in the same way as explained in Remark 6.4.9.

6.5.1 The family of lower bound graphs

We begin with describing a fixed graph construction, $F = (V_F, E_F, w_F)$, and then we describe how to get from F and a vector of strings $\bar{x} \in \prod_{i=1}^{t}\{0,1\}^{k^2}$ the graph $F_{\bar{x}} = (V, E_{\bar{x}}, x_{\bar{x}})$. Let G be the fixed graph construction defined in Section 6.4.1. The fixed graph construction F consists of exactly two copies of G, denoted by G^1 and G^2. Recall that $G = (V_G, E_G)$ where $V_G = \bigcup_{i=1}^{t} A^i \cup Code^i$. In order to distinguish between the sets of nodes that belong to G^1 and the sets of nodes that belong to G^2, we add an ordered pair as a superscript (i,b), where $b \in \{1,2\}$ indicates whether the set is in G^1 or in G^2. That is, the set of nodes of G^1 is $V_{G^1} = \bigcup_{i=1}^{t} A^{(i,1)} \cup Code^{(i,1)}$, and the set of nodes of G^2 is $V_{G^2} = \bigcup_{i=1}^{t} A^{(i,2)} \cup Code^{(i,2)}$. Hence the set of nodes of F is $V_F = \bigcup_{i=1}^{t} V^i$, where for any $i \in [t]$, we denote by

$$\begin{aligned}
V^i &= V^{(i,2)} \cup V^{(i,2)} \\
V^{(i,1)} &= A^{(i,1)} \cup Code^{(i,1)} \\
V^{(i,2)} &= A^{(i,2)} \cup Code^{(i,2)} \\
A^{(i,1)} &= \{v_m^{(i,1)} \mid m \in [k]\} \\
A^{(i,2)} &= \{v_m^{(i,2)} \mid m \in [k]\} \\
Code^{(i,1)} &= \bigcup_{h=1}^{\ell+\alpha} C_h^{(i,1)} \\
Code^{(i,2)} &= \bigcup_{h=1}^{\ell+\alpha} C_h^{(i,2)} \\
C_h^{(i,1)} &= \{\sigma_{(h,1)}^{(i,1)}, \cdots, \sigma_{(h,\ell+\alpha)}^{(i,1)}\} \\
C_h^{(i,2)} &= \{\sigma_{(h,1)}^{(i,2)}, \cdots, \sigma_{(h,\ell+\alpha)}^{(i,2)}\} \\
Code_w^{(i,1)} &= \{\sigma_{(h,w_h)}^{(i,1)} \mid h \in [\ell+\alpha]\} \\
Code_w^{(i,2)} &= \{\sigma_{(h,w_h)}^{(i,2)} \mid h \in [\ell+\alpha]\}
\end{aligned}$$

The weight function w_F is defined as follows. For any $v \in V_F$,

$$w_F(v) = \begin{cases} \ell & \text{if } v \in \bigcup_{i=1}^{t} A^{(i,1)} \cup A^{(i,2)} \\ 1 & \text{otherwise} \end{cases}$$

That is, the weight of any node in the cliques $\bigcup_{i=1}^{t} A^{(i,1)} \cup A^{(i,2)}$ is ℓ, and the weight of any node in the code-gadgets $\bigcup_{i=1}^{t} Code^{(i,1)} \cup Code^{(i,2)}$ is 1. Observe that unlike the previous section, the weights of the nodes don't depend on the strings in $\bar{x}$.

Obtaining $F_{\bar{x}}$ from $F = (V_F, E_F, w_F)$ and $\bar{x}$. Let $\bar{x} = (x^1, \cdots, x^t) \in \prod_{i=1}^{t} \{0,1\}^{k^2}$. For any x^i, we index the k^2 positions in x^i by $x^i_{(m_1,m_2)}$, for $m_1, m_2 \in [k]$. The graph $F_{\bar{x}}$ is defined as follows. The set of nodes and the weight function remain exactly as in F. The set of edges contains all the edges in F, and the following edges in $A^{(i,1)} \times A^{(i,2)}$, for any $i \in [t]$.

$$\{v^{(i,1)}_{m_1}, v^{(i,2)}_{m_2} \mid x^i_{(m_1,m_2)} = 0\}$$

That is, for any $i \in [t]$ and any $m_1, m_2 \in [k]$, we add an edge between $v^{(i,1)}_{m_1} \in A^{(i,1)}$ and $v^{(i,2)}_{m_2} \in A^{(i,2)}$ if and only if $x^i_{(m_1,m_2)} = 0$.

6.5.2 $F_{\bar{x}}$ is a $(3/4+\epsilon)$-approximate $MaxIS$ family of lower bound graphs

In this section we prove the following lemma.

Lemma 6.5.1. *For any constant $\epsilon > 0$, there is a constant $t > 2$ for which it holds that $\{ F_{\bar{x}} \mid \bar{x} \in \prod_{i=1}^{t} \{0,1\}^{k^2} \}$ is a $(3/4+\epsilon)$-approximate MaxIS family of lower bound graphs.*

Lemma 6.5.1 is a corollary of Claims 6.5.2 and 6.5.3.

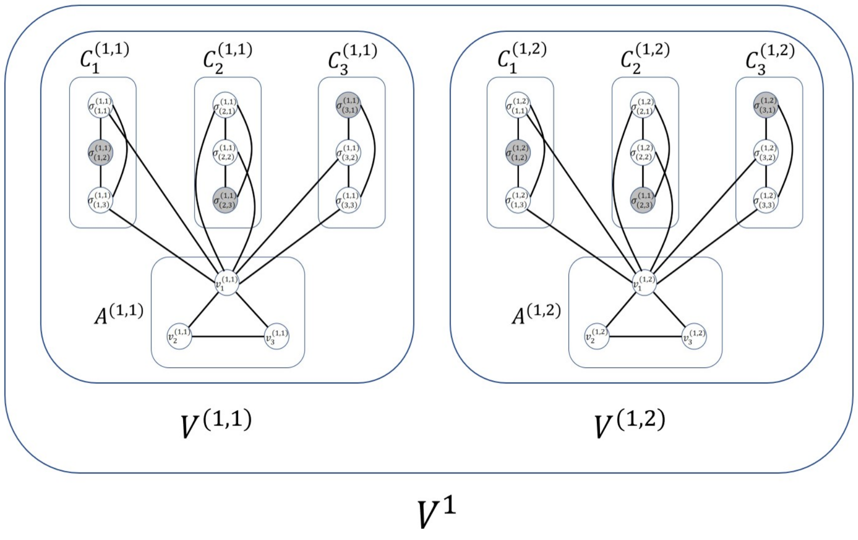

Figure 6.3: An example of the graph induced by the nodes in V^1. As in the previous figures, $\ell = 2, \alpha = 1$ and $k = (\ell+\alpha)^\alpha = 3$. V^1 contains two sets of nodes: $V^{(1,1)}$ which is in G^1, and $V^{(1,2)}$ which is in G^2. The graph induced by the nodes in $V^{(1,1)}$ has an identical topology to the graph induced by the nodes in $V^{(1,2)}$, and they are both identical to the topology of the base graph construction H that was described is Section 6.4.1. The reason that there is an ordered pair $(1, b)$, where $b \in \{1, 2\}$ in a superscript in $V^{(1,1)}$ and $V^{(1,2)}$ is as follows. The first element in the pair indicates that these sets are parts of V^1, and the second element b in the pair indicates that $V^{(1,b)}$ belongs to G^b. Similarly, $V^{(1,1)} = A^{(1,1)} \cup Code^{(1,1)} = A^{(1,1)} \cup C_1^{(1,1)} \cup C_2^{(1,1)} \cup C_3^{(1,1)}$, and $V^{(1,2)} = A^{(1,2)} \cup Code^{(1,2)} = A^{(1,2)} \cup C_1^{(1,2)} \cup C_2^{(1,2)} \cup C_3^{(1,2)}$. As in the previous figures, the code-mapping of 1, $\mathcal{C}(1) =$ "2, 3, 1", and therefore, $v_1^{(1,1)}$ is connected to all the nodes in $Code^{(1,1)}$ except of the nodes in $Code_1^{(1,1)} = \{\sigma_{(1,2)}^{(1,1)}, \sigma_{(2,3)}^{(1,1)}, \sigma_{(3,1)}^{(1,1)}\}$. Similarly, $v_1^{(1,2)}$ is connected to all the nodes in $Code^{(1,2)}$ except of the nodes in $Code_1^{(1,2)} = \{\sigma_{(1,2)}^{(1,2)}, \sigma_{(2,3)}^{(1,2)}, \sigma_{(3,1)}^{(1,2)}\}$. Some edges are omitted in this figure, for clarity.

Claim 6.5.2. For any $g_{\bar{x}} \in \{ F_{\bar{x}} \mid \bar{x} \in \prod_{i=1}^{t} \{0,1\}^{k^2} \}$, if there is a pair $(m_1, m_2) \in [k] \times [k]$ for which it holds that $x^1_{(m_1,m_2)} = x^2_{(m_1,m_2)} = \cdots = x^t_{(m_1,m_2)} = 1$, then $g_{\bar{x}}$ contains an independent set of weight at least $4t\ell + 2\alpha t$.

Proof. Consider the following set of nodes.

$$I = \bigcup_{i=1}^{t} \{v^{(i,1)}_{m_1}\} \cup Code^{(i,1)}_{m_1} \cup \{v^{(i,2)}_{m_2}\} \cup Code^{(i,2)}_{m_2}$$

First, by Property 3, it holds that both $\bigcup_{i=1}^{t} \{ v^{(i,1)}_{m_1} \} \cup Code^{(i,1)}_{m_1}$ and $\bigcup_{i=1}^{t} \{ v^{(i,2)}_{m_2} \} \cup Code^{(i,2)}_{m_2}$ are independent sets. Furthermore, the only possible edges between $\bigcup_{i=1}^{t} \{ v^{(i,1)}_{m_1} \} \cup Code^{(i,1)}_{m_1}$ and $\bigcup_{i=1}^{t} \{ v^{(i,2)}_{m_2} \} \cup Code^{(i,2)}_{m_2}$ are the ones in $\{\{v^{(i,1)}_{m_1}, v^{(i,2)}_{m_2}\} \mid i \in [t]\}$. But since $x^1_{(m_1,m_2)} = x^2_{(m_1,m_2)} = \cdots = x^t_{(m_1,m_2)} = 1$, none of the edges in $\{\{v^{(i,1)}_{m_1}, v^{i_2}_{m_2}\} \mid i \in [t]\}$ exists in the graph $F_{\bar{x}}$. The weight of I is

$$|\bigcup_{i=1}^{t} w(\{v^{(i,1)}_{m_1}, v^{(i,2)}_{m_2}\})| + |\bigcup_{i=1}^{t} w(Code^{(i,1)}_{m_1} \cup Code^{(i,2)}_{m_2})| = 2t\ell + 2t(\ell + \alpha) = t(4\ell + \alpha)$$

as desired. □

Claim 6.5.3. For any $g_{\bar{x}} \in \{ G_{\bar{x}} \mid \bar{x} \in \prod_{i=1}^{t} \{0,1\}^{k^2} \}$, if the strings $x^1, x^2, \cdots, x^t$ are pairwise disjoint, then the weight of any independent set in $g_{\bar{x}}$ is at most $3(t+1)\ell + 3\alpha t^3$.

Proof. The proof is by induction on t. For $t = 1$, observe that $w(I \cap V^1) = w(I \cap (V^{(1,1)} \cup V^{(1,2)})) = w(I \cap (A^{(1,1)} \cup Code^{(1,1)} \cup A^{(1,2)} \cup Code^{(1,2)})) \leqslant 4\ell + 2\alpha$. We assume correctness for $t-1$, and we prove correctness for t. Let I be an independent set in $g_{\bar{x}}$. The proof is by the following case analysis.

1. There is some $i \in [t]$, for which it holds that $|I \cap (A^{(i,1)} \cup A^{(i,2)})| \leqslant 1$: In this case, observe that $w(I \cap V^i) = w(I \cap (A^{(i,1)} \cup A^{(i,2)} \cup Code^{(i,1)} \cup Code^{(i,2)})) = w(I \cap (A^{(i,1)} \cup A^{(i,2)})) + w(I \cap (Code^{(i,1)} \cup Code^{(i,2)})) \leqslant \ell + 2(\ell + \alpha)$. Hence, by applying the inductive hypothesis on the graph induced by the nodes in $\bigcup_{j \in [t] \setminus \{i\}} V^j$, we deduce that $w(I) = w(I \cap \bigcup_{j \in [t] \setminus \{i\}} V^j) + w(I \cap V^i) \leqslant 3t\ell + 3\alpha(t-1)^3 + 3\ell + 2\alpha < 3(t+1)\ell + 3\alpha t^3$.

2. For all $i \in [t]$, it holds that $|I \cap (A^{(i,1)} \cup A^{(i,2)})| = 2$: This case is proved without applying the inductive hypothesis, as follows. Fist, since $A^{(i,1)}$ and $A^{(i,2)}$ are cliques, there is one node in $I \cap A^{(i,1)}$ and one node in $I \cap A^{(i,2)}$. Denote these two nodes by $v^{(i,1)}_{m^1_i} \in A^{(i,1)}$ and $v^{(i,2)}_{m^2_i} \in A^{(i,2)}$, where $m^1_i, m^2_i \in [k]$. This implies that $v^{(i,1)}_{m^1_i}$ and $v^{(i,2)}_{m^2_i}$ are not connected by an edge. Since the strings $x^1, \cdots, x^t$ are pairwise disjoint, it must be the case that all the pairs in $\{(m^1_i, m^2_i) \mid i \in [t]\}$ are distinct.

   We split the multiset of indices $\{m^1_i \mid i \in [t]\}$ into *equivalence classes* by their value, where each class contains a set of indices of the same value. Observe that there are positive integers $r, q_1, q_2, \cdots, q_r$ satisfying $\sum_{j=1}^{r} q_j = t$, for which we can split $\{m^1_i \mid i \in [t]\}$ into r equivalence classes $Q_1, \cdots, Q_r$, where $|Q_j| = q_j$. Let $s_i = \sum_{j=1}^{i} q_j$.[5] Assume without loss of generality that

$$\begin{aligned}
&Q_1 = \{m^1_1, \cdots, m^1_{s_1}\}\\
&Q_2 = \{m^1_{s_1+1}, \cdots, m^1_{s_2}\}\\
&Q_3 = \{m^1_{s_2+1}, \cdots, m^1_{s_3}\}\\
&\vdots\\
&Qr = \{m^1_{s_{r-1}+1}, \cdots, m^1_{t}\}
\end{aligned}$$

   where

$$\begin{aligned}
&m^1_1 = \cdots = m^1_{s_1}\\
&m^1_{s_1+1} = \cdots = m^1_{s_2}\\
&m^1_{s_2+1} = \cdots = m^1_{s_3}\\
&\vdots\\
&m^1_{s_{r-1}+1} = \cdots = m^1_{t}
\end{aligned}$$

   That is, we are assuming without loss of generality that Q_1 contains the first $s_1 = q_1$ indices in $\{m^1_i \mid i \in [t]\}$, Q_2 contains the next q_2 indices in $\{m^1_i \mid i \in$

[5]For example, if the multiset is $\{1, 1, 2, 3, 3, 3, 5\}$, then we have $r = 4, q_1 = 2, q_2 = 1, q_3 = 3, q_4 = 1$

$[t]\}$, etc. This assumption is indeed without loss of generality because we can always split $\{m_i^1 \mid i \in [t]\}$ into r equivalence classes by their values, for some positive integer r, and our proof doesn't depend on the actual elements in each class. Since the pairs in $\{(m_i^1, m_i^2) \mid i \in [t]\}$ are distinct, it must be the case that

$$\begin{aligned}
&m_1^2 \neq \cdots \neq m_{s_1}^2 \\
&m_{s_1+1}^2 \neq \cdots \neq m_{s_2}^2 \\
&m_{s_2+1}^2 \neq \cdots \neq m_{s_3}^2 \\
&\vdots \\
&m_{s_{r-1}+1}^2 \neq \cdots \neq m_t^2
\end{aligned}$$

The idea of the proof is to split the set of nodes into 3 disjoint sets, where the intersection of the independent set with each of the sets has small weight, as follows (we set $s_0 = 0$).

$$\begin{aligned}
V &= \bigcup_{i=1}^{t} V^{(i,1)} \cup V^{(i,2)} \\
&= \overbrace{\left(\bigcup_{j=0}^{r-1} V^{(s_j+1,1)})\right)}^{\text{First set}} \cup \overbrace{\left(\bigcup_{j=1}^{r}(\bigcup_{i=s_{j-1}+2}^{s_j} V^{(i,1)})\right)}^{\text{Second set}} \cup \overbrace{\left(\bigcup_{j=1}^{r}(\bigcup_{i=s_{j-1}+1}^{s_j} V^{(i,2)})\right)}^{\text{Third set}}
\end{aligned}$$

In Propositions 6.5.4, 6.5.5, and 6.5.6, we show that the intersection of the independent set with each of the three sets has small weight, and therefore, in total, the weight of the independent set is sufficiently small.

Proposition 6.5.4. *It holds that*

$$w\left(I \cap (\bigcup_{j=0}^{r-1} V^{(s_j+1,1)})\right) \leqslant (r+1)\ell + \alpha t^2$$

Proof. Since $m^1_1, m^1_{s_1+1}, \cdots, m^1_{s_{r-1}+1}$ are in different equivalence classes, they are distinct. Hence, by applying Corollary 6.4.7, we have that

$$w\left(I \cap (\bigcup_{j=0}^{r-1} V^{(s_j+1,1)})\right) \leqslant (r+1)\ell + \alpha r^2 \leqslant (r+1)\ell + \alpha t^2$$

□

Proposition 6.5.5. *It holds that*

$$w\left(I \cap \bigcup_{j=1}^{r} (\bigcup_{i=s_{j-1}+2}^{s_j} V^{(i,1)})\right) \leqslant 2\ell(t-r) + \alpha(t-r)$$

Proof. Since for any $i \in [t]$, $A^{(i,1)}$ is a clique, and $Code^{(i,1)}$ is a union of $\ell + \alpha$ cliques, we have that

$$\begin{aligned}
& w\left(\bigcup_{j=1}^{r} (\bigcup_{i=s_{j-1}+2}^{s_j} V^{(i,1)})\right) = \sum_{j=1}^{r} \sum_{i=s_{j-1}+2}^{s_j} w(I \cap V^{(i,1)}) \\
& = \sum_{j=1}^{r} \sum_{i=s_{j-1}+2}^{s_j} w\left(I \cap (A^{(i,1)} \cup Code^{(i,1)})\right) \\
& \leqslant \sum_{j=1}^{r} \sum_{i=s_{j-1}+2}^{s_j} 2\ell + \alpha \\
& = \sum_{j=1}^{r} 2\ell(|Q_j| - 1) + \alpha(|Q_j| - 1) \\
& = \sum_{j=1}^{r} 2\ell(q_j - 1) + \alpha(q_j - 1) \\
& = 2\ell(\sum_{j=1}^{r} q_j) + 2\ell(\sum_{j=1}^{r} -1) + \alpha(\sum_{j=1}^{r} q_j - 1) \\
& \leqslant 2\ell(t-r) + \alpha(t-r)
\end{aligned}$$

where the final inequality holds because $\sum_{j=1}^{r} q_j = t$. □

Proposition 6.5.6. *It holds that*

$$w\left(I \cap \bigcup_{j=1}^{r}(\bigcup_{i=s_{j-1}+1}^{s_j} V^{(i,2)})\right) \leqslant (t+r)\ell + \alpha t^3$$

Proof. Since for any $j \in [r]$, it holds that $m^2_{s_{j-1}+1} \neq \cdots \neq m^2_{s_j}$. We can apply Corollary 6.4.7 on the graph induced by the nodes in $\bigcup_{i=s_{j-1}+1}^{s_j} V^{(i,2)}$ to deduce that

$$\begin{aligned} &w(I \cap (\bigcup_{i=s_{j-1}+1}^{s_j} V^{(i,2)})) \leqslant (|Q_j|+1)\ell + \alpha(|Q_j|)^2 \\ &= (q_j+1)\ell + \alpha {q_j}^2 \end{aligned}$$

Hence, we have that

$$\begin{aligned} &w\left(I \cap \bigcup_{j=1}^{r}(\bigcup_{i=s_{j-1}+1}^{s_j} V^{(i,2)})\right) = \sum_{j=1}^{r} w(I \cap (\bigcup_{i=s_{j-1}+1}^{s_j} V^{(i,2)})) \leqslant \sum_{j=1}^{r} (q_j+1)\ell + \alpha q_j^2 \\ &\leqslant (t+r)\ell + \alpha t^2 r \leqslant (t+r)\ell + \alpha t^3 \end{aligned}$$

□

In total, we have that

$$\begin{aligned} w(I) = w&\left(I \cap (\bigcup_{j=0}^{r-1} V^{(s_j+1,1)})\right) \\ &+ w\left(I \cap \bigcup_{j=1}^{r}(\bigcup_{i=s_{j-1}+2}^{s_j} V^{(i,1)})\right) \\ &+ w\left(I \cap \bigcup_{j=1}^{r}(\bigcup_{i=s_{j-1}+1}^{s_j} V^{(i,2)})\right) \end{aligned}$$

Hence,

$$\begin{aligned}w(I) &\leqslant (r+1)\ell + \alpha t^2 + 2\ell(t-r) + \alpha(t-r) + (t+r)\ell + \alpha t^3 \\ &\leqslant \ell(r+1+2t-2r+t+r) + 3\alpha t^3 \\ &= \ell(3t+1) + 3\alpha t^3\end{aligned}$$

as desired.

□

Proof of Lemma 6.5.1. Claims 6.5.2 and 6.5.3 imply that the family of graphs $\{ G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t} \{0,1\}^{k^2} \}$ is a family of lower bound graphs with respect to the pairwise disjointness function and the graph predicate that distinguishes between graphs of Maximum Independent Set at least $4t\ell + 2\alpha t$ and graphs of Maximum Independent Set at most $3(t+1)\ell + 3\alpha t^3$.

Recall that $\ell = \log k - \log k / \log\log k, \alpha = \log k / \log\log k$. Which implies that the graph predicate distinguishes between independent sets of weight at least

$$\begin{aligned}&4t(\log k - \log k/\log\log k) + 2\log k/\log\log k \\ &= 4t\log k - 2t\log k/\log\log k \geqslant 4(t-1)\log k\end{aligned}$$

and independent sets of weight at most

$$3(t+1)(\log k - \log k/\log\log k) + t^2(\log k/\log\log k) \leqslant 3(t+2)\log k$$

, for any constant t and $k \gg t$. Hence, for any constant $\epsilon > 0$, we choose $t = (3/4\epsilon) - 1$ (or the first integer larger than $t = (3/4\epsilon) - 1$, if it is not an integer). This implies that for any constant $0 < \epsilon \leqslant 1/4$, there is a constant t for which $\{ G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t} \{0,1\}^{k^2} \}$ is a $(3/4+\epsilon)$-approximate *MaxIS* family of graphs. □

Proof of Theorem 6.1.2. Observe that $k = \Theta(tn) = \Theta(n)$, where $n = |V|$. Furthermore, by Lemma 6.5.1, $\{ G_{\bar{x}} = (V, E_{\bar{x}}, w_{\bar{x}}) \mid \bar{x} \in \prod_{i=1}^{t} \{0,1\}^{k^2} \}$ is a $(3/4+\epsilon)$-approximate *MaxIS* family of graphs, where the partition of the set of nodes that is needed for Definition 6.3.1 is $V = \bigcup_{i=1}^{t} V^i$. Hence, by Corollary 6.3.5, the fact that the length of the strings is $k^2 = \Theta(n^2)$, and the fact that $|cut(G_{\bar{x}})| =$

$\Theta(t^2 \log^2 k) = \Theta(\log^2 k)$, any algorithm for finding a $(3/4+\epsilon)$-approximation for Maximum Independent Set in the CONGEST model with success probability at least 2/3 requires $\Omega(k^2/(t \log t \cdot |cut(G_{\bar{x}})| \log |V|)) = \Omega(n^2/(t \log t \cdot \log^3 n) = \Omega(n^2/\log^3 n)$ rounds. $\square$

Bibliography

[1] Amir Abboud, Raghavendra Addanki, Fabrizio Grandoni, Debmalya Panigrahi, and Barna Saha. Dynamic set cover: improved algorithms and lower bounds. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 114–125, 2019. 17

[2] Amir Abboud and Arturs Backurs. Towards hardness of approximation for polynomial time problems. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. 32

[3] Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. *J. ACM*, 64(4):28:1–28:20, 2017. 9

[4] Amir Abboud, Greg Bodwin, and Seth Pettie. A hierarchy of lower bounds for sublinear additive spanners. *SIAM J. Comput.*, 47(6):2203–2236, 2018. 9, 32

[5] Amir Abboud, Karl Bringmann, and Nick Fischer. Stronger 3-sum lower bounds for approximate distance oracles via additive combinatorics. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 391–404. ACM, 2023. 10

[6] Amir Abboud, Karl Bringmann, Seri Khoury, and Or Zamir. Hardness of approximation in p via short cycle removal: cycle detection, distance oracles, and beyond. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1487–1500. ACM, 2022. 6, 16

[7] Amir Abboud, Keren Censor-Hillel, and Seri Khoury. Near-linear lower bounds for distributed distance computations, even in sparse networks. In *Distributed Computing - 30th International Symposium, DISC*, volume 9888 of *Lecture Notes in Computer Science*, pages 29–42. Springer, 2016. 11, 58, 59, 60, 61, 64, 207

[8] Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Christoph Lenzen. Fooling views: a new lower bound technique for distributed computations under congestion. *Distributed Comput.*, 33(6):545–559, 2020. 26

[9] Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Ami Paz. Smaller cuts, higher lower bounds. *ACM Trans. Algorithms*, 17(4), oct 2021. 2

[10] Amir Abboud and Aviad Rubinstein. Fast and deterministic constant factor approximation algorithms for LCS imply new circuit lower bounds. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. 32

[11] Amir Abboud, Aviad Rubinstein, and Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 25–36. IEEE, 2017. 17, 18

[12] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 434–443. IEEE, 2014. 2, 18, 51

[13] Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 377–391. SIAM, 2016. 32

[14] Udit Agarwal and Vijaya Ramachandran. Fine-grained complexity for sparse graphs. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 239–252. ACM, 2018. 32

[15] Gagan Aggarwal, Rajeev Motwani, Devavrat Shah, and An Zhu. Switch scheduling via randomized edge coloring. In *FOCS*, page 502, 2003. 124

[16] Mohamad Ahmadi and Fabian Kuhn. Distributed maximum matching verification in CONGEST. In *DISC*, volume 179 of *LIPIcs*, pages 37:1–37:18, 2020. 123

[17] Mohamad Ahmadi, Fabian Kuhn, and Rotem Oshman. Distributed approximate maximum matching in the CONGEST model. In *DISC*, volume 121 of *LIPIcs*, pages 6:1–6:17, 2018. 123

[18] Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999. 32

[19] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM, 2021. 21

[20] Noga Alon. A simple algorithm for edge-coloring bipartite multigraphs. *Information Processing Letters*, 85(6):301–302, March 2003. 124

[21] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7(4):567–583, 1986. 12

[22] Noga Alon, Sonny Ben-Shimon, and Michael Krivelevich. A note on regular ramsey graphs. *J. Graph Theory*, 64(3):244–249, 2010. 124

[23] Noga Alon, Amin Coja-Oghlan, Hiêp Hàn, Mihyun Kang, Vojtech Rödl, and Mathias Schacht. Quasi-randomness and algorithmic regularity for graphs with general degree distributions. *SIAM J. Comput.*, 39(6):2336–2362, 2010. 124

[24] Noga Alon, Shmuel Friedland, and Gil Kalai. Every 4-regular graph plus an edge contains a 3-regular subgraph. *J. Comb. Theory B*, 37(1):92–93, 1984. 124

[25] Noga Alon, Shmuel Friedland, and Gil Kalai. Regular subgraphs of almost regular graphs. *J. Comb. Theory B*, 37(1):79–91, 1984. 124

[26] Noga Alon and Guy Moshkovitz. Limitations on regularity lemmas for clustering graphs. *Adv. Appl. Math.*, 124:102135, 2021. 124

[27] Noga Alon and Pawel Pralat. Modular orientations of random and quasi-random regular graphs. *Comb. Probab. Comput.*, 20(3):321–329, 2011. 124

[28] Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *SODA*, pages 1132–1139. SIAM, 2012. 127

[29] Noga Alon and Joel Spencer. *The Probabilistic Method*. John Wiley, 1992. 83, 92

[30] Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997. 19, 24

[31] Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993. 9, 31

[32] Bertie Ancona, Keren Censor-Hillel, Mina Dalirrooyfard, Yuval Efron, and Virginia Vassilevska Williams. Distributed distance approximation. In Quentin Bramas, Rotem Oshman, and Paolo Romano, editors, *24th International Conference on Principles of Distributed Systems, OPODIS 2020, December 14-16, 2020, Strasbourg, France (Virtual Conference)*, volume 184 of *LIPIcs*, pages 30:1–30:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. 59, 62

[33] Moab Arar, Shiri Chechik, Sarel Cohen, Cliff Stein, and David Wajc. Dynamic matching: Reducing integral algorithms to approximately-maximal fractional algorithms. In *ICALP*, volume 107 of *LIPIcs*, pages 7:1–7:16, 2018. 123

[34] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. 210, 211

[35] Armen S. Asratian and Nikolai N. Kuzjurin. On the number of nearly perfect matchings in almost regular uniform hypergraphs. *Discret. Math.*, 207(1-3):1–8, 1999. 127

[36] Sepehr Assadi, Soheil Behnezhad, Sanjeev Khanna, and Huan Li. On regularity lemma and barriers in streaming and dynamic matching. In *STOC*, pages 131–144, 2023. 123

[37] Sepehr Assadi and Janani Sundaresan. Hidden permutations to the rescue: Multi-pass streaming lower bounds for approximate matchings. In *FOCS*, pages 909–932, 2023. 123

[38] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Near-linear time construction of sparse neighborhood covers. *SIAM Journal on Computing*, 28(1):263–277, 1998. 7, 17

[39] Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. Network decomposition and locality in distributed computation. In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 364–369. IEEE Computer Society, 1989. 12

[40] László Babai. On the complexity of canonical labeling of strongly regular graphs. *SIAM J. Comput.*, 9(1):212–216, 1980. 124

[41] László Babai. On the automorphism groups of strongly regular graphs I. In *ITCS*, pages 359–368. ACM, 2014. 124

[42] László Babai, Xi Chen, Xiaorui Sun, Shang-Hua Teng, and John Wilmes. Faster canonical forms for strongly regular graphs. In *FOCS*, pages 157–166. IEEE Computer Society, 2013. 124

[43] Nir Bachrach, Keren Censor-Hillel, Michal Dory, Yuval Efron, Dean Leitersdorf, and Ami Paz. Hardness of distributed optimization. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019.*, pages 238–247, 2019. 2, 60, 82, 207, 208, 210

[44] Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Toward tight approximation bounds for graph diameter and eccentricities. *SIAM Journal on Computing*, 50(4):1155–1199, 2021. 9, 17

[45] Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. On acyclic conjunctive queries and constant delay enumeration. In *International Workshop on Computer Science Logic*, pages 208–222. Springer, 2007. 26

[46] Egon Balas and Chang Sung Yu. Finding a maximum clique in an arbitrary graph. *SIAM J. Comput.*, 15(4):1054–1068, 1986. 12

[47] Alkida Balliu, Thomas Boudier, Sebastian Brandt, and Dennis Olivetti. Tight lower bounds in the supported LOCAL model. In *PODC*, pages 95–105. ACM, 2024. 125, 129

[48] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 481–497. IEEE Computer Society, 2019. 12, 14

[49] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. *J. ACM*, 68(5):39:1–39:30, 2021. 125, 129

[50] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Improved distributed lower bounds for MIS and bounded (out-)degree dominating sets in trees. In *PODC*, pages 283–293. ACM, 2021. 125, 129

[51] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Distributed Δ-coloring plays hide-and-seek. In *STOC*, pages 464–477. ACM, 2022. 125, 129

[52] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Distributed maximal matching and maximal independent set on hypergraphs. In *SODA*, pages 2632–2676. SIAM, 2023. 125, 129

[53] Alkida Balliu, Sebastian Brandt, and Dennis Olivetti. Distributed lower bounds for ruling sets. *SIAM J. Comput.*, 51(1):70–115, 2022. 125, 129

[54] Nikhil Bansal and Ryan Williams. Regularity lemmas and combinatorial algorithms. *Theory Comput.*, 8(1):69–94, 2012. 28

[55] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM (JACM)*, 48(5):1069–1090, 2001. 84, 87

[56] Reuven Bar-Yehuda, Keren Censor-Hillel, Mohsen Ghaffari, and Gregory Schwartzman. Distributed approximation of maximum independent set and maximum matching. In *PODC*, pages 165–174. ACM, 2017. 82, 83, 87, 88, 123, 124

[57] Reuven Bar-Yehuda, Keren Censor-Hillel, and Gregory Schwartzman. A distributed (2+ϵ)-approximation for vertex cover in o(logδ/ϵ log log δ) rounds. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC*, pages 3–8, 2016. 124

[58] Reuven Bar-Yehuda, Keren Censor-Hillel, and Gregory Schwartzman. A distributed (2 + ϵ)-approximation for vertex cover in o(log Δ / ϵ log log Δ) rounds. *J. ACM*, 64(3):23:1–23:11, 2017. 87

[59] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. 1

[60] Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2013. 12

[61] Leonid Barenboim, Michael Elkin, and Fabian Kuhn. Distributed (delta+1)-coloring in linear (in delta) time. *SIAM J. Comput.*, 43(1):72–95, 2014. 12

[62] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 321–330, 2012. 12

[63] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *J. ACM*, 63(3):20:1–20:45, 2016. 12, 82

[64] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *J. ACM*, 63(3):20:1–20:45, 2016. 124

[65] Surender Baswana, Akshay Gaur, Sandeep Sen, and Jayant Upadhyay. Distance oracles for unweighted graphs: Breaking the quadratic barrier with constant additive error. In *International Colloquium on Automata, Languages, and Programming*, pages 609–621. Springer, 2008. 7, 17

[66] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (α, β)-spanners. *ACM Transactions on Algorithms (TALG)*, 7(1):1–26, 2010. 32

[67] Surender Baswana, Sumeet Khurana, and Soumojit Sarkar. Fully dynamic randomized algorithms for graph spanners. *ACM Transactions on Algorithms (TALG)*, 8(4):1–51, 2012. 17

[68] Surender Baswana and Sandeep Sen. Approximate distance oracles for unweighted graphs in expected $o(n^2)$ time. *ACM Transactions on Algorithms (TALG)*, 2(4):557–577, 2006. 7, 17

[69] Soheil Behnezhad, Mohammad Roghani, and Aviad Rubinstein. Local computation algorithms for maximum matching: New lower bounds. In *FOCS*, pages 2322–2335. IEEE, 2023. 128

[70] Ran Ben-Basat, Guy Even, Ken-ichi Kawarabayashi, and Gregory Schwartzman. Optimal distributed covering algorithms. *Distributed Comput.*, 36(1):45–55, 2023. 130, 150

[71] Ran Ben-Basat, Ken-ichi Kawarabayashi, and Gregory Schwartzman. Parameterized distributed algorithms. In *DISC*, 2019. 123, 135, 188, 189

[72] Clark T. Benson. Minimal regular graphs of girths eight and twelve. *Canadian Journal of Mathematics*, 18:1091–1094, 1966. 64

[73] Thiago Bergamaschi, Monika Henzinger, Maximilian Probst Gutenberg, Virginia Vassilevska Williams, and Nicole Wein. New techniques and fine-grained hardness for dynamic near-additive spanners. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1836–1855. SIAM, 2021. 32

[74] Aaron Bernstein. Fully dynamic (2+ε) approximate all-pairs shortest paths with fast query and close to linear update time. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 693–702. IEEE, 2009. 17

[75] Aaron Bernstein and Liam Roditty. Improved dynamic algorithms for maintaining approximate shortest paths under deletions. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1355–1365. SIAM, 2011. 17

[76] Sayan Bhattacharya, Peter Kiss, and Thatchaphol Saranurak. Dynamic (1+ϵ)-approximate matching size in truly sublinear update time. In *FOCS*, pages 1563–1588, 2023. 123

[77] Sayan Bhattacharya, Peter Kiss, Thatchaphol Saranurak, and David Wajc. Dynamic matching with better-than-2 approximation in polylogarithmic update time. In *SODA*, pages 100–128, 2023. 123

[78] Etienne Birmelé, Rui Ferreira, Roberto Grossi, Andrea Marino, Nadia Pisanti, Romeo Rizzi, and Gustavo Sacomoto. Optimal listing of cycles and st-paths in undirected graphs. In *Proceedings of the twenty-fourth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1884–1896. SIAM, 2013. 23

[79] G.A. Bodino. *Economic applications of the theory of graphs*. Tracts on mathematics and its applications. Gordon and Breach, Science Publishers, 1962. 12

[80] Marijke HL Bodlaender, Magnús M Halldórsson, Christian Konrad, and Fabian Kuhn. Brief announcement: Local independent set approximation. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, To Appear 2016. 82

[81] Immanuel M. Bomze, Marco Budinich, Panos M. Pardalos, and Marcello Pelillo. The maximum clique problem. In *Handbook of Combinatorial Optimization*, pages 1–74. Kluwer Academic Publishers, 1999. 12

[82] John A Bondy and Miklós Simonovits. Cycles of even length in graphs. *Journal of Combinatorial Theory, Series B*, 16(2):97–105, 1974. 19

[83] Édouard Bonnet. 4 vs 7 sparse undirected unweighted diameter is SETH-hard at time $n^{4/3}$. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 34:1–34:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 17

[84] Édouard Bonnet. Inapproximability of diameter in super-linear time: Beyond the 5/3 ratio. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 17:1–17:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 17

[85] Ravi B. Boppana, Magnús M. Halldórsson, and Dror Rawitz. Simple and local independent set approximation. In *SIROCCO*, volume 11085 of *Lecture Notes in Computer Science*, pages 88–101. Springer, 2018. 83

[86] Jean Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985. 17, 31

[87] Sebastian Brandt. An automatic speedup theorem for distributed problems. In *PODC*, pages 379–388. ACM, 2019. 125, 129

[88] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A lower bound for the distributed lovász local lemma. In *STOC*, pages 479–488. ACM, 2016. 125, 129

[89] Sebastian Brandt and Dennis Olivetti. Truly tight-in-Δ bounds for bipartite maximal matching and variants. In Yuval Emek and Christian Cachin, editors, *PODC*, pages 69–78. ACM, 2020. 125, 129

[90] Karl Bringmann and Sebastian Krinninger. Brief announcement: A note on hardness of diameter approximation. In *31st International Symposium on Distributed Computing, DISC*, volume 91 of *LIPIcs*, pages 44:1–44:3. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. 61

[91] Karl Bringmann, Marvin Künnemann, and Karol Wegrzycki. Approximating APSP without scaling: equivalence of approximate min-plus and exact min-max. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 943–954. ACM, 2019. 32

[92] Niv Buchbinder, Joseph (Seffi) Naor, and David Wajc. Lossless online rounding for online bipartite matching (despite its impossibility). In *SODA*, pages 2030–2068, 2023. 123

[93] Marc Bury, Elena Grigorescu, Andrew McGregor, Morteza Monemizadeh, Chris Schwiegelshohn, Sofya Vorotnikova, and Samson Zhou. Structural results on matching estimation with applications to streaming. *Algorithmica*, 81(1):367–392, 2019. 123

[94] S. Butenko and W. E. Wilhelm. Clique-detection models in computational biochemistry and genomics. *European Journal of Operational Research*, 173:1–17, 2005. 12

[95] Sergiy Butenko, Panos M. Pardalos, Ivan Sergienko, Vladimir Shylo, and Petro Stetsyuk. Finding maximum independent sets in graphs arising from coding theory. In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC), March 10-14, 2002, Madrid, Spain*, pages 542–546, 2002. 12

[96] Nofar Carmeli and Markus Kröll. On the enumeration complexity of unions of conjunctive queries. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 134–148, 2019. 23, 51

[97] Nofar Carmeli and Markus Kröll. Enumeration complexity of conjunctive queries with functional dependencies. *Theory of Computing Systems*, 64(5):828–860, 2020. 23

[98] Teena Carroll, David J. Galvin, and Prasad Tetali. Matchings and independent sets of a fixed size in regular graphs. *J. Comb. Theory A*, 116(7):1219–1227, 2009. 124

[99] Keren Censor-Hillel and Michal Dory. Distributed spanner approximation. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Com-*

puting, PODC 2018, Egham, United Kingdom, July 23-27, 2018, pages 139–148. ACM, 2018. 60, 207

[100] Keren Censor-Hillel, Michal Dory, Janne H. Korhonen, and Dean Leitersdorf. Fast approximate shortest paths in the congested clique. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC*, pages 74–83. ACM, 2019. 11, 58

[101] Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic methods in the congested clique. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC*, pages 143–152. ACM, 2015. 11, 58

[102] Keren Censor-Hillel, Telikepalli Kavitha, Ami Paz, and Amir Yehudayoff. Distributed construction of purely additive spanners. In *Proceedings of the 30th International Symposium on Distributed Computing, DISC*, pages 129–142, 2016. 60

[103] Keren Censor-Hillel, Seri Khoury, and Ami Paz. Quadratic and near-quadratic lower bounds for the CONGEST model. In *31st International Symposium on Distributed Computing, DISC*, pages 10:1–10:16, 2017. 60, 63, 64, 82, 207, 211, 212

[104] Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman. Derandomizing local distributed algorithms under bandwidth restrictions. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, volume 91 of *LIPIcs*, pages 11:1–11:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. 82

[105] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark*, pages 107–117, 2003. 209, 210

[106] Timothy M. Chan, Virginia Vassilevska Williams, and Yinzhan Xu. Algorithms, reductions and equivalences for small weight variants of all-pairs shortest paths. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming,*

ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference), volume 198 of *LIPIcs*, pages 47:1–47:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 54

[107] Yi-Jun Chang, Seth Pettie, Thatchaphol Saranurak, and Hengjie Zhang. Near-optimal distributed triangle enumeration via expander decompositions. *J. ACM*, 68(3):21:1–21:36, 2021. 28

[108] Shiri Chechik. Near-optimal approximate decremental all pairs shortest paths. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 170–181. IEEE, 2018. 17, 18

[109] Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert Endre Tarjan, and Virginia Vassilevska Williams. Better approximation algorithms for the graph diameter. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1041–1052. SIAM, 2014. 32

[110] Shiri Chechik, Yang P Liu, Omer Rotem, and Aaron Sidford. Constant girth approximation for directed graphs in subquadratic time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1010–1023, 2020. 32

[111] Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. *Theory Comput.*, 16:1–50, 2020. 17

[112] Lijie Chen, Shafi Goldwasser, Kaifeng Lyu, Guy N Rothblum, and Aviad Rubinstein. Fine-grained complexity meets IP=PSPACE. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–20. SIAM, 2019. 17, 32

[113] Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 21–40. SIAM, 2019. 17

[114] Fan R. K. Chung and Lincoln Lu. Survey: Concentration inequalities and martingale inequalities: A survey. *Internet Math.*, 3(1):79–127, 2006. 138, 139

[115] Edith Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM Journal on Computing*, 28(1):210–236, 1998. 7, 17

[116] Edith Cohen and Uri Zwick. All-pairs small-stretch paths. *Journal of Algorithms*, 38(2):335–353, 2001. 7, 17

[117] Ilan Reuven Cohen and David Wajc. Randomized online matching in regular graphs. In *SODA*, pages 960–979, 2018. 124

[118] Richard Cole and John E. Hopcroft. On edge coloring bipartite graphs. *SIAM J. Comput.*, 11(3):540–546, 1982. 124

[119] Artur Czumaj and Christian Konrad. Detecting cliques in CONGEST networks. In *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, volume 121 of *LIPIcs*, pages 16:1–16:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. 60, 207

[120] Andrzej Czygrinow and Michal Hanckowiak. Distributed algorithm for better approximation of the maximum matching. In *Computing and Combinatorics, 9th Annual International Conference, COCOON 2003, Big Sky, MT, USA, July 25-28, 2003, Proceedings*, pages 242–251, 2003. 123

[121] Andrzej Czygrinow, Michal Hańćkowiak, and Wojciech Wawrzyniak. Fast distributed approximations in planar graphs. In *Distributed Computing*, pages 78–92. Springer, 2008. 82

[122] Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Morten Stöckel. Finding even cycles faster via capped k-walks. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 112–120. ACM, 2017. 23, 24, 26, 32

[123] Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Morten Stöckel. New subquadratic approximation algorithms for the girth. *arXiv preprint arXiv:1704.02178*, 2017. 17

[124] Elias Dahlhaus and Marek Karpinski. Perfect matching for regular graphs is AC°-hard for the general matching problem. *J. Comput. Syst. Sci.*, 44(1):94–102, 1992. 124

[125] Mina Dalirrooyfard, Ray Li, and Virginia Vassilevska Williams. Hardness of approximate diameter: Now for undirected graphs. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1021–1032. IEEE, 2021. 17

[126] Mina Dalirrooyfard and Nicole Wein. Tight conditional lower bounds for approximating diameter in directed graphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1697–1710, 2021. 17

[127] Mina Dalirrooyfard and Virginia Vassilevska Williams. Conditionally optimal approximation algorithms for the girth of a directed graph. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 35:1–35:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. 32

[128] Mina Dalirrooyfard, Virginia Vassilevska Williams, Nikhil Vyas, and Nicole Wein. Tight approximation algorithms for bichromatic graph diameter and related problems. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 47:1–47:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. 17

[129] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. *Engineering Route Planning Algorithms*, pages 117–139. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. 1

[130] Camil Demetrescu and Giuseppe F Italiano. A new approach to dynamic all pairs shortest paths. *Journal of the ACM (JACM)*, 51(6):968–992, 2004. 17

[131] Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000. 7, 17, 32

[132] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In Magnús M. Halldórsson and Shlomi Dolev, editors, *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 367–376. ACM, 2014. 31, 60, 64, 207

[133] Guillaume Ducoffe. Faster approximation algorithms for computing shortest cycles on weighted graphs. *SIAM Journal on Discrete Mathematics*, 35(2):953–969, 2021. 17

[134] Bartlomiej Dudek and Pawel Gawrychowski. Computing quartet distance is equivalent to counting 4-cycles. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 733–743. ACM, 2019. 32

[135] Bartlomiej Dudek and Pawel Gawrychowski. Counting 4-patterns in permutations is equivalent to counting 4-cycles in graphs. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPIcs*, pages 23:1–23:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. 32

[136] Lech Duraj, Krzysztof Kleiner, Adam Polak, and Virginia Vassilevska Williams. Equivalences between triangle and range query problems. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 30–47. SIAM, 2020. 51

[137] Arnaud Durand and Etienne Grandjean. First-order queries on structures of bounded degree are computable with constant delay. *ACM Transactions on Computational Logic (TOCL)*, 8(4):21–es, 2007. 23

[138] David A. Easley and Jon M. Kleinberg. *Networks, Crowds, and Markets - Reasoning About a Highly Connected World.* Cambridge University Press, 2010. 1

[139] Yuval Efron, Ofer Grossman, and Seri Khoury. Beyond alice and bob: Improved inapproximability for maximum independent set in CONGEST. In Yuval Emek and Christian Cachin, editors, *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event*, pages 511–520. ACM, 2020. 13, 207

[140] Michael Elkin. An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. *SIAM J. Comput.*, 36(2):433–456, 2006. 60, 207

[141] David Eppstein, Zvi Galil, Giuseppe F Italiano, and Amnon Nissenzweig. Sparsification—a technique for speeding up dynamic graph algorithms. *Journal of the ACM (JACM)*, 44(5):669–696, 1997. 1, 4

[142] Paul Erdös. On some extremal problems in graph theory. *Israel Journal of Mathematics*, 3(2):113–116, 1965. 31

[143] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8:128–140, 1736. 1

[144] Guy Even, Moti Medina, and Dana Ron. Distributed maximum matching in bounded degree graphs. In *ICDCN*, pages 18:1–18:10, 2015. 123

[145] Salwa Faour, Mohsen Ghaffari, Christoph Grunau, Fabian Kuhn, and Václav Rozhon. Local distributed rounding: Generalized to mis, matching, set cover, and beyond. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 4409–4447. SIAM, 2023. 82

[146] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *SIAM J. Comput.*, 38(5):1709–1727, 2008. 6

[147] Manuela Fischer. Improved deterministic distributed matching via rounding. *Distributed Comput.*, 33(3-4):279–291, 2020. 123

[148] Manuela Fischer, Mohsen Ghaffari, and Fabian Kuhn. Deterministic distributed edge-coloring via hypergraph maximal matching. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 180–191, 2017. 12

[149] Manuela Fischer, Slobodan Mitrovic, and Jara Uitto. Deterministic $(1+\epsilon)$-approximate maximum matching with $\mathrm{poly}(1/\epsilon)$ passes in the semi-streaming model and beyond. In *STOC*, pages 248–260, 2022. 123, 124, 125, 128

[150] Orr Fischer, Tzlil Gonen, Fabian Kuhn, and Rotem Oshman. Possibilities and impossibilities for distributed subgraph detection. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA*, pages 153–162. ACM, 2018. 60, 207

[151] Abraham D. Flaxman and Shlomo Hoory. Maximum matchings in regular graphs of high girth. *Electron. J. Comb.*, 14(1), 2007. 124

[152] Fernando Florenzano, Cristian Riveros, Martín Ugarte, Stijn Vansummeren, and Domagoj Vrgoc. Constant delay algorithms for regular document spanners. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 165–177, 2018. 23

[153] Sebastian Forster, Gramoz Goranci, and Monika Henzinger. Dynamic maintenance of low-stretch probabilistic tree embeddings with applications. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1226–1245. SIAM, 2021. 17, 18

[154] 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. IEEE Computer Society, 2018. 62

[155] Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. Networks cannot compute their diameter in sublinear time. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1150–1162, 2012. 11, 58, 59, 60, 64, 207

[156] Zoltán Füredi and Miklós Simonovits. *The History of Degenerate (Bipartite) Extremal Graph Problems*, pages 169–264. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. 60, 64

[157] Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 270–277. SIAM, 2016. 12, 82

[158] Mohsen Ghaffari. Distributed maximal independent set using small messages. In *SODA*, pages 805–820. SIAM, 2019. 12, 82

[159] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrovic, and Ronitt Rubinfeld. Improved massively parallel computation algorithms for mis, matching, and vertex cover. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 129–138, 2018. 12

[160] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrovic, and Ronitt Rubinfeld. Improved massively parallel computation algorithms for MIS, matching, and vertex cover. In *PODC*, pages 129–138. ACM, 2018. 124, 125, 130, 153

[161] Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In *FOCS*, pages 662–673, 2018. 123

[162] Mohsen Ghaffari and Fabian Kuhn. Distributed minimum cut approximation. In *Distributed Computing - 27th International Symposium, DISC 2013, Jerusalem, Israel, October 14-18, 2013. Proceedings*, volume 8205 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2013. 60, 207

[163] Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 784–797, 2017. 82

[164] Mohsen Ghaffari, Fabian Kuhn, Yannic Maus, and Jara Uitto. Deterministic distributed edge-coloring with fewer colors. In *STOC*, pages 418–430, 2018. 123

[165] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings in O(n log n) time in regular bipartite graphs. *SIAM Journal on Computing*, 42(3):1392–1404, 2013. 123, 124

[166] Shafi Goldwasser and Ofer Grossman. Bipartite perfect matching in pseudo-deterministic NC. In *ICALP*, volume 80 of *LIPIcs*, pages 87:1–87:13, 2017. 124

[167] Louis Golowich. A new berry-esseen theorem for expander walks. In *STOC*, pages 10–22. ACM, 2023. 124

[168] Tzlil Gonen and Rotem Oshman. Lower bounds for subgraph detection in the CONGEST model. In *21st International Conference on Principles of Distributed Systems, OPODIS 2017, Lisbon, Portugal, December 18-20, 2017*, volume 95 of *LIPIcs*, pages 6:1–6:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. 60, 207

[169] Fabrizio Grandoni, Stefano Leonardi, Piotr Sankowski, Chris Schwiegelshohn, and Shay Solomon. $(1 + \epsilon)$-approximate incremental matching in constant deterministic amortized time. In *SODA*, pages 1886–1898, 2019. 123

[170] Ofer Grossman, Seri Khoury, and Ami Paz. Improved hardness of approximation of diameter in the CONGEST model. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPIcs*, pages 19:1–19:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. 11, 31, 58

[171] Anupam Gupta, Guru Guruganesh, Binghui Peng, and David Wajc. Stochastic online metric matching. In *ICALP*, volume 132 of *LIPIcs*, pages 67:1–67:14, 2019. 123

[172] Manoj Gupta and Richard Peng. Fully dynamic $(1+\epsilon)$-approximate matchings. In *FOCS*, pages 548–557, 2013. 123

[173] Magnús M. Halldórsson and Christian Konrad. Computing large independent sets in a single round. *Distributed Computing*, 31(1):69–82, 2018. 82

[174] David G. Harris. Distributed local approximation algorithms for maximum matching in graphs and hypergraphs. In *FOCS*, pages 700–724, 2019. 123, 124, 126, 128, 130, 137, 143

[175] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Decremental single-source shortest paths on undirected graphs in near-linear total update time. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 146–155. IEEE, 2014. 17

[176] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic

problems via the online matrix-vector multiplication conjecture. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 21–30, 2015. 17, 18

[177] Monika R Henzinger and Valerie King. Maintaining minimum spanning trees in dynamic graphs. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 194–203. ACM, 1995. 1, 4

[178] Stephan Holzer, David Peleg, Liam Roditty, and Roger Wattenhofer. Distributed 3/2-approximation of the diameter. In *Proceedings of the 28th International Symposium on Distributed Computing, DISC*, pages 562–564, 2014. 11, 58, 59, 60, 61

[179] Stephan Holzer and Nathan Pinsker. Approximation of distances and shortest paths in the broadcast congest clique. In *19th International Conference on Principles of Distributed Systems, OPODIS*, volume 46 of *LIPIcs*, pages 6:1–6:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. 11, 58, 62, 64

[180] Stephan Holzer and Nathan Pinsker. Approximation of distances and shortest paths in the broadcast congest clique. In *Proceedings of the 19th International Conference on Principles of Distributed Systems, OPODIS*, pages 6:1–6:16, 2015. 60, 207

[181] Stephan Holzer and Roger Wattenhofer. Optimal distributed all pairs shortest paths and applications. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC*, pages 355–364, 2012. 11, 58, 59, 61

[182] Shlomo Hoory. *On graphs of high girth*. PhD thesis, Citeseer, 2002. 31

[183] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973. 123

[184] Qiang-Sheng Hua, Haoqiang Fan, Lixiang Qian, Ming Ai, Yangyang Li, Xuanhua Shi, and Hai Jin. Brief announcement: A tight distributed algorithm for all pairs shortest paths and applications. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA*, pages 439–441, 2016. 11, 58, 59

[185] Amos Israeli and Alon Itai. A fast and simple randomized parallel algorithm for maximal matching. *Inf. Process. Lett.*, 22(2):77–80, 1986. 123, 125

[186] Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing*, 7(4):413–423, 1978. 17

[187] Taisuke Izumi, Naoki Kitamura, and Yutaro Yamaguchi. A nearly linear-time distributed algorithm for exact maximum matching. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 4062–4082. SIAM, 2024. 123

[188] Svante Janson. Large deviations for sums of partly dependent random variables. *Random Structures & Algorithms*, 24(3):234–248, 2004. 137

[189] Ce Jin and Yinzhan Xu. Removing additive structure in 3sum-based reductions. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 405–418. ACM, 2023. 10

[190] Avi Kadaria, Liam Roditty, Aaron Sidford, Virginia Vassilevska Williams, and Uri Zwick. Algorithmic trade-offs for girth approximation in undirected graphs. In *SODA*, 2022. 17, 18

[191] Jeff Kahn and Jeong Han Kim. Random matchings in regular graphs. *Comb.*, 18(2):201–226, 1998. 124

[192] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *STOC*, pages 587–596, 2011. 123

[193] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972. 2

[194] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, pages 352–358, 1990. 123

[195] CS Karthik and Pasin Manurangsi. On closest pair in euclidean metric: Monochromatic is as hard as bichromatic. *Combinatorica*, 40(4):539–573, 2020. 17

[196] Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5):33:1–33:38, 2019. 17

[197] Ken-ichi Kawarabayashi, Seri Khoury, Aaron Schild, and Gregory Schwartzman. Improved distributed approximations for maximum independent set. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPIcs*, pages 35:1–35:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. 13, 82, 89

[198] Seri Khoury, Manish Purohit, Aaron Schild, and Joshua Wang. Distributed approximate maximum matching in regular graphs. In *Under Submission*, 2024. 14, 123

[199] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017. 1

[200] Jon M. Kleinberg. The small-world phenomenon: an algorithmic perspective. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 163–170. ACM, 2000. 1

[201] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1272–1287. SIAM, 2016. 8, 20, 33

[202] Ludek Kucera. Canonical labeling of regular graphs in linear average time. In *FOCS*, pages 271–279. IEEE Computer Society, 1987. 124

[203] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 980–989, 2006. 130

[204] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *J. ACM*, 63(2):17:1–17:44, 2016. 12, 13, 83, 124, 126

[205] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 1997. 62

[206] Christoph H. Lampert, Liva Ralaivola, and Alexander Zimin. Dependency-dependent bounds for sums of dependent random variables, 2018. 137

[207] Christoph Lenzen and David Peleg. Efficient distributed source detection with limited bandwidth. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC*, pages 375–382, 2013. 11, 58

[208] Christoph Lenzen and Roger Wattenhofer. Leveraging linial's locality limit. In *Distributed Computing*, pages 394–407. Springer, 2008. 82

[209] Christoph Lenzen and Roger Wattenhofer. MIS on trees. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 41–48, 2011. 12

[210] Ray Li. Settling SETH vs. approximate sparse directed unweighted diameter (up to (NU)NSETH). In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1684–1696, 2021. 17

[211] Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1236–1252. SIAM, 2018. 32

[212] Andrzej Lingas and Eva-Marta Lundell. Efficient approximation algorithms for shortest cycles in undirected graphs. In *Latin American Symposium on Theoretical Informatics*, pages 736–746. Springer, 2008. 17

[213] Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992. 1, 3, 12, 90, 91, 125

[214] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. *J. ACM*, 62(5):38:1–38:17, 2015. 123, 124

[215] Zvi Lotker, Boaz Patt-Shamir, and Adi Rosén. Distributed approximate matching. *SIAM J. Comput.*, 39(2):445–460, 2009. 123

[216] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986. 12, 13, 83

[217] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986. 127, 131

[218] Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *FOCS*, pages 253–262, 2013. 123

[219] Jiří Matoušek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel Journal of Mathematics*, 93(1):333–344, 1996. 17, 31

[220] Colin McDiarmid. *On the method of bounded differences*, page 148–188. London Mathematical Society Lecture Note Series. Cambridge University Press, 1989. 152

[221] Andrew McGregor. Finding graph matchings in data streams. In *APPROX*, volume 3624 of *Lecture Notes in Computer Science*, pages 170–181, 2005. 123

[222] Aranyak Mehta. Online matching and ad allocation. *Found. Trends Theor. Comput. Sci.*, 8(4):265–368, 2013. 123

[223] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge International Series on Parallel Computation. Cambridge University Press, 1995. 124

[224] Danupon Nanongkai, Atish Das Sarma, and Gopal Pandurangan. A tight unconditional lower bound on distributed randomwalk computation. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 257–266. ACM, 2011. 60, 207

[225] Moni Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM J. Discret. Math.*, 4(3):409–412, 1991. 12, 91, 114, 125

[226] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003. 1

[227] Jakub Pachocki, Liam Roditty, Aaron Sidford, Roei Tov, and Virginia Vassilevska Williams. Approximating cycles in directed graphs: Fast algorithms for girth and roundtrip spanners. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1374–1392. SIAM, 2018. 32

[228] Alessandro Panconesi and Romeo Rizzi. Some simple distributed algorithms for sparse networks. *Distributed Computing*, 14(2):97–100, 2001. 12

[229] Alessandro Panconesi and Aravind Srinivasan. On the complexity of distributed network decomposition. *J. Algorithms*, 20(2):356–374, 1996. 12

[230] Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theor. Comput. Sci.*, 381(1-3):183–196, 2007. 128

[231] Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610. ACM, 2010. 8, 10, 20, 22, 33, 53

[232] Mihai Patrascu and Liam Roditty. Distance oracles beyond the Thorup-Zwick bound. *SIAM J. Comput.*, 43(1):300–311, 2014. The conference version appeared in FOCS 2010. 7, 9, 17, 18, 21

[233] Mihai Patrascu, Liam Roditty, and Mikkel Thorup. A new infinity of distance oracles for sparse graphs. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 738–747. IEEE Computer Society, 2012. 7, 9, 17, 18, 19, 21

[234] Ami Paz and Gregory Schwartzman. A $(2+\epsilon)$-approximation for maximum weight matching in the semi-streaming model. *ACM Trans. Algorithms*, 15(2):18:1–18:15, 2019. 123

[235] David Peleg. *Distributed Computing: A Locality-sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000. 1, 3

[236] David Peleg, Liam Roditty, and Elad Tal. Distributed algorithms for network diameter and girth. In *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 660–672, 2012. 11, 58

[237] David Peleg and Vitaly Rubinovich. A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction. *SIAM J. Comput.*, 30(5):1427–1442, 2000. 2, 60, 207

[238] David Peleg and Alejandro A Schäffer. Graph spanners. *Journal of graph theory*, 13(1):99–116, 1989. 9, 31

[239] Sriram V. Pemmaraju. Equitable coloring extends chernoff-hoeffding bounds. In Michel X. Goemans, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques, 4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2001*, volume 2129, pages 285–296. Springer, 2001. 89

[240] Seth Pettie and Peter Sanders. A simpler linear time $2/3 - \epsilon$ approximation for maximum weight matching. *Information Processing Letters*, 91(6):271–276, 2004. 137

[241] Jeff M. Phillips, Elad Verbin, and Qin Zhang. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 486–501, 2012. 209

[242] M.D. Plummer and L. Lovász. *Matching Theory*. ISSN. Elsevier Science, 1986. 136

[243] Liam Roditty and Virginia Vassilevska Williams. Subquadratic time approximation algorithms for the girth. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 833–845. SIAM, 2012. 17, 32

[244] Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In Dan Boneh, Tim

Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 515–524. ACM, 2013. 32

[245] Liam Roditty and Uri Zwick. On dynamic shortest paths problems. In *European Symposium on Algorithms*, pages 580–591. Springer, 2004. 18

[246] Liam Roditty and Uri Zwick. Dynamic approximate all-pairs shortest paths in undirected graphs. *SIAM Journal on Computing*, 41(3):670–683, 2012. 17

[247] Václav Rozhon and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 350–363. ACM, 2020. 12, 82, 85

[248] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *Innovations in Computer Science (ICS)*, pages 223–238. Tsinghua University Press, 2011. 127

[249] Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1260–1268, 2018. 17

[250] Aviad Rubinstein and Virginia Vassilevska Williams. SETH vs approximation. *ACM SIGACT News*, 50(4):57–76, 2019. 2, 17, 20

[251] Peter Sanders and Dominik Schultes. Highway hierarchies hasten exact shortest path queries. In Gerth Stølting Brodal and Stefano Leonardi, editors, *Algorithms – ESA 2005*, pages 568–579, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. 1

[252] 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. *SIAM J. Comput.*, 41(5):1235–1265, 2012. 2, 60, 207

[253] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. 1

[254] Georg Schnitger and Bala Kalyanasundaram. The probabilistic communication complexity of set intersection. In *Proceedings of the Second Annual Conference on Structure in Complexity Theory, Cornell University, Ithaca, New York, USA, June 16-19, 1987*, pages 41–47. IEEE Computer Society, 1987. 63

[255] Luc Segoufin. Constant delay enumeration for conjunctive queries. *ACM SIGMOD Record*, 44(1):10–17, 2015. 23

[256] Roded Sharan and Trey Ideker. Modeling cellular machinery through biological network comparison. *Nature Biotechnology*, 24(4):427–433, April 2006. Funding Information: R.S. is supported by an Alon Fellowship; T.I., by the David and Lucille Packard Foundation. This work was also supported by the National Center for Research Resources (RR018627) and the National Science Foundation (NSF 0425926). 1

[257] Yossi Shiloach and Shimon Even. An on-line edge-deletion problem. *J. ACM*, 28(1):1–4, jan 1981. 1, 4

[258] Yossi Shiloach and Shimon Even. An on-line edge-deletion problem. *Journal of the ACM (JACM)*, 28(1):1–4, 1981. 17

[259] R. Singleton. On minimal graphs of maximum even girth. *Journal of Combinatorial Theory*, 1:306–332, 12 1966. 64

[260] Shay Solomon. Fully dynamic maximal matching in constant update time. In *FOCS*, pages 325–334, 2016. 123

[261] Christian Sommer, Elad Verbin, and Wei Yu. Distance oracles for sparse graphs. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 703–712. IEEE, 2009. 7, 17, 18, 21, 31

[262] Daniel A. Spielman. Faster isomorphism testing of strongly regular graphs. In *STOC*, pages 576–584. ACM, 1996. 124

[263] Terence Tao. Structure and randomness in combinatorics. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 3–15. IEEE, 2007. 28

[264] Mikkel Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM (JACM)*, 46(3):362–394, 1999. 7, 17

[265] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1):1–24, 2005. 7, 17, 31

[266] Jan van den Brand, Yin Tat Lee, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Bipartite matching in nearly-linear time on moderately dense graphs. In *FOCS*, pages 919–930, 2020. 123

[267] Virginia Vassilevska Williams. Hardness of easy problems: basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 43, 2015. 33, 51

[268] Virginia Vassilevska Williams and Yinzhan Xu. Monochromatic triangles, triangle listing and APSP. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 786–797. IEEE, 2020. 8, 20, 33, 51

[269] Zhaocai Wang, Jian Tan, Lanwei Zhu, and Wei Huang. Solving the maximum independent set problem based on molecule parallel supercomputing. *Applied Mathematics and Information Sciences*, 8:2361–2366, 09 2014. 12

[270] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3447–3487. World Scientific, 2018. 33, 51

[271] Virginia Vassilevska Williams and R Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *Journal of the ACM (JACM)*, 65(5):1–38, 2018. 53

[272] Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. 53

[273] David P. Woodruff. Lower bounds for additive spanners, emulators, and more. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 389–398. IEEE Computer Society, 2006. 9

[274] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, STOC*, pages 209–213, 1979. 62

[275] Raphael Yuster. Maximum matching in regular and almost regular graphs. *Algorithmica*, 66(1):87–92, 2013. 123, 124

[276] Raphael Yuster and Uri Zwick. Finding even cycles even faster. *SIAM J. Discret. Math.*, 10(2):209–222, 1997. The conference version appeared in ICALP 1994. 19

[277] Or Zamir. Algorithmic applications of hypergraph and partition containers. In *STOC*, pages 985–998. ACM, 2023. 124

[278] Bin Zhang and Steve Horvath. A general framework for weighted gene co-expression network analysis. *Statistical Applications in Genetics and Molecular Biology*, 4(1), 2005. 1