Streaming Maximum Matching in a Few Passes: Algorithms and Lower Bounds

By

Kheeran K. Naidu

School of Computer Science
University of Bristol

May 2024

A dissertation submitted to the University of Bristol in accordance with the requirements for award of the degree of *Doctor of Philosophy* in the Faculty of Engineering.

Word count: 60,000

Abstract

We study the approximate maximum bipartite matching problem in the insertion-only streaming model. The simple $(1/2 = 0.5)$-approximate Greedy maximal matching algorithm is the best-known one-pass algorithm using $O(n \operatorname{polylog} n)$ (semi-streaming) space where n is the number of vertices in the graph. Doing better remains an elusive open question and has thus motivated its study in a few passes, where improvements have been made.

Considering first the class of deterministic Greedy-only algorithms, we give a $(5/8 = 0.625)$-approximate three-pass semi-streaming algorithm. This improves on the previous best arbitrary three-pass semi-streaming algorithm by Feldman and Szarf [APPROX'22]. We show that any Greedy-only algorithm requires three passes to do better than a 1/2-approximation and that our three-pass algorithm is optimal, thus completing our understanding of Greedy-only algorithms up to three passes.

Almost all two-pass algorithms fall into the class of Greedy-first algorithms, including the current best $(2 - \sqrt{2} \approx 0.585)$-approximate semi-streaming algorithm by Konrad [MFCS'18]. We obtain a meta-algorithm that yields this algorithm and the deterministic algorithm by Esfandiari, Hajiaghayi, and Monemizadeh [ICDM'16] as special cases, thus unifying two strands of research. It also yields a new $(2 - \sqrt{2})$-approximate algorithm. We further show that any two-pass Greedy-first algorithm with a better than $(2/3 \approx 0.667)$-approximation requires more than semi-streaming space, narrowing the approximation gap to $[2 - \sqrt{2}, 2/3]$.

As our main contribution, we give the first unconditional space lower bound for arbitrary two-pass algorithms. We show that any such algorithm with a better than $(8/9 \approx 0.889)$-approximation requires strictly more than semi-streaming space. This improves on the previous conditional result by Assadi [SODA'22], even under its best conditions. Therefore, we narrow the approximation factor gap of two-pass semi-streaming algorithms to $[2 - \sqrt{2}, 8/9]$.

Finally, we show that the Greedy maximal matching algorithm is also a powerful subroutine in the insertion-deletion model and give an α-approximation algorithm for minimum vertex cover using $O(n^2/\alpha^2)$ space, matching the space lower bound by Dark and Konrad [CCC'20] up to constant factors.

Dedication and Acknowledgements

I am truly grateful for all the things that I have learned, all the places that I have been, all the people that I have met, and all the experiences that I have had over the past four years of my PhD. Due to space constraints, in the following, I acknowledge only some of all that I am grateful for.

My growing home, which now crosses city, country, and continental boarders, has been my solid core and something that I am lucky to have: My parents who laid the foundations for everything that I have accomplished since before I can remember. My three brothers who have been consistent sources of laughter, adventure, and reflection throughout my life. My nephew and niece who continue to make my world better and brighter – an extra thank you to my brother and his partner for these bundles of joy. And, my loving partner who has literally been by my side throughout the entire course of my PhD.

My family and friends, regardless of where in the world, have also been strong pillars of support throughout my PhD and even prior to it. Whether I was in Bristol, London or Malaysia, there has always been a good group of people that I can depend on and celebrate with – including those who I have not seen in years, but catchup with on calls lasting several hours. I am grateful to have such amazing people in my life.

My academic home at the Algorithms and Complexity group in the University of Bristol has been a warm and welcoming one. My supervisor, Christian Konrad, has been an excellent mentor and collaborator. Our meetings and conversations – since the very first one that introduced me to the one-pass streaming maximum matching problem – have undoubtedly had a huge impact on the kind of researcher that I am today. I am extremely grateful for all that he has done. Additionally, I am grateful for all the postdocs and students who are or have been in the group. From our meetings and conversations on interesting research questions to our hikes of the surrounding areas in all sorts of weather, our joint experiences have been invaluable.

I am also fortunate to have had the opportunity to meet and collaborate with excellent researchers, both peers and more established ones. Keen insights, helpful guidance, and many in-depth discussions are just some of the things that I am grateful for. A special thank you to Sepehr Assadi for the opportunity to work closely with him and his PhD students, Vihan Shah and Janani Sundaresan.

Lastly, I am especially grateful for all the interactions that I have had with members of the Theoretical Computer Science community at conferences, research visits, and other events. Regardless of experience or stature, everyone has been extremely approachable and easy to talk to. A special shout out to everyone who has been apart of the many memorable days and nights at these events.

Thank you, everyone!

Author's Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's *Regulations and Code of Practice for Research Degree Programmes* and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE: .....

Table of Contents

List of Tables and Figures

1 Introduction

Algorithms for processing massive datasets are in high demand due to the exponential growth rate of data being created and stored [Tay23]. In particular, graph datasets, where entities are represented as *vertices* and relations between them are represented as *edges*, are becoming more prevalent. This includes graphs for social networks, the internet map, and network traffic data that, in many cases, are already too large to store on the working memory (random-access memory) of a computer, namely, they are *massive graphs*. As a concrete example of this, the Facebook social network, with users as vertices and friendships as edges, is a massive graph that is known to be on the scale of trillions of edges [CEK+15].

Graph datasets are typically analysed and studied by solving fundamental graph problems (e.g. [Kyp21]). However, these problems have traditionally been solved in the *random-access machine* (RAM) model of computation where an algorithm is assumed to have access to any edge of the graph in constant time (see [CLRS22] for examples). This implies that the entire graph needs to be stored on the working memory of a computer, which is an infeasible assumption when considering massive graphs. To combat this problem, models of computation that do not rely on this assumption have been developed and studied (e.g. [Lin87, AMS96, DG04]). In this thesis, we focus on one such model for processing massive graphs, the *streaming* model of computation.

1.1 The Graph Streaming Model

The streaming model was first formalised by Alon, Matias, and Szegedy [AMS96] for processing massive datasets. In the context of graphs, this gives us the *graph streaming model* for processing massive graphs where the edges of an n-vertex graph are sequentially presented to an algorithm as a *stream* of edges in any fixed arbitrary order. A trivial algorithm for any problem can simply store all the edges of the stream using $O(n^2)$ space and then run any traditional (RAM model) algorithm to solve the problem. Therefore, the goal of a streaming algorithm is to solve the problem using space $o(n^2)$, that is, space sublinear in the number of edges.

Research in this area began with *insertion-only* streams where the stream is a sequence of

edge insertions that make up the edge set of the underlying input graph. In their seminal works [FKM+04, FKM+05], Feigenbaum et al. initiated the study of *semi-streaming* algorithms that are allowed $\tilde{O}(n) = O(n \operatorname{polylog} n)$ space and aim to use the fewest number of passes of the stream. They tackle the problem of finding a maximum matching (MM), that is, the largest set of vertex-disjoint edges in the input graph. In fact, semi-streaming space is a natural space regime for MM since just storing the edges of a solution already requires $\Omega(n \log n)$ bits of space in graphs where the maximum matching is of size $\Theta(n)$. They further justify the semi-streaming space regime by showing that, even for testing basic properties of graphs such as connectivity and bipartiteness, $\Omega(n)$ space is necessary[1] and $O(n \log n)$ space is sufficient in one pass of the stream.

The semi-streaming space regime has subsequently received significant attention and, for many graph problems, semi-streaming algorithms that provably use the fewest number of passes of the stream are now known, even when considering *insertion-deletion* streams where edges that are inserted may later be deleted. See Table 1.1 for a summary of some of these problems. Semi-streaming algorithms have also been studied in the context of *random-order* and *sliding-window* streams, and the interested reader may refer to [CMS13, McG14, Kon18b, Ber20, AB21, ADKN23] and the references therein for further details. In contrast, before the introduction of the semi-streaming space regime, very few graph problems had been studied in the streaming model – see the survey by Muthukrishnan [Mut05] for more details.

Over the past two decades, studying the fewest number of passes achievable by a semi-streaming algorithm for a given graph problem has provided an understanding of its hardness in the streaming model. Consider, for example, the maximum clique, maximal independent set, and minimum cut problems in insertion-only streams. Although they are known to require $\Omega(n^2)$ space in one pass [HSSW12, BLS+18, CDK19, ACK19b, Zel11], their hardness varies significantly in multiple passes. Minimum cut turns out to be a comparatively easy problem as it already admits a semi-streaming algorithm in just two passes of the stream [RSW18, AD21]. For maximal independent set, it was very recently shown that any semi-streaming algorithm requires $\Omega(\log \log n)$ passes [AKNS24], thus matching the long-standing $O(\log \log n)$-pass semi-streaming algorithm[2] [ACG+15] (see also [GGK+18, Kon18a]). On the other hand, maximum clique is a very difficult problem for semi-streaming algorithms since even constant factor approximations require $\Omega(n/\operatorname{polylog} n)$ many passes [HSSW12, BLS+18].

Understanding the hardness of a graph problem in this way has thus been a fundamental question in the streaming model. Despite the considerable progress that has been made for some problems, for many other graph problems, it is still unclear exactly how difficult they are, i.e., how well the problem can be solved using semi-streaming space. To that end, in this thesis, we make contributions towards answering this question for *approximate maximum*

[1] The space lower bounds for testing connectivity and bipartiteness were later improved to $\Omega(n \log n)$ [SW15].
[2] This algorithm also applies to insertion-deletion streams.

Problem	Model	# Passes	Space
connectivity & bipartitness	insertion-only	1	$\Theta(n \log n)$ [FKM+04, SW15]
	insertion-deletion	1	$\tilde{\Theta}(n)$ [AGM12a, SW15]
k-edge-connectivity	insertion-only	1	$\Omega(kn)$ [SW15]
	insertion-deletion	1	$\tilde{O}(kn)$ [AGM12a]
spanning forest	insertion-only	1	$\Theta(n \log n)$ [FKM+04, SW15]
	insertion-deletion	1	$\Theta(n \log^3 n)$ [AGM12a, NY19]
minimum spanning tree	insertion-only	1	$\Theta(n \log n)$ [FKM+04, SW15]
	insertion-deletion	$p \geq 1$	$n^{1+\Theta(1/p)}$ [AGM12a, AKZ24]
$(1+\varepsilon)$-min spanning tree	insertion-only	≥ 1	$\Omega(n)$ [trivial]
	insertion-deletion	1	$\tilde{O}(n/\varepsilon)$ [AGM12a]
minimum cut	insertion-only	1	$\Omega(n^2)$ [Zel11]
	insertion-only	≥ 2	$\tilde{\Theta}(n)$ [AD21]
$(1+\varepsilon)$-minimum cut	insertion-only	1	$\Omega(n \log(n/\varepsilon))$ [AG09, SW15]
	insertion-deletion	1	$\tilde{O}(n/\varepsilon^2)$ [AGM12b]
$(1 \pm \varepsilon)$-spectral sparsification	insertion-only	≥ 1	$\Omega(n)$ [trivial]
	insertion-deletion	1	$O(n \log n/\varepsilon^2)$ [KLM+14]
vertex colouring	insertion-only	≥ 1	$\Omega(n)$ [trivial]
	insertion-deletion	1	$\tilde{O}(n)$ [ACK19b, AKM22]
maximal independent set*	insertion-only	$p \geq 1$	$n^{1+\Omega(1/2^p)-o(1)}$ [AKNS24]
	insertion-deletion	$p \geq 1$	$n^{1+O(1/2^p)}$ [ACG+15]

Table 1.1: A summary of several graph problems with semi-streaming algorithms that use exactly the optimal number of passes. Problems indicated by an asterisk (*) have semi-streaming algorithms that are known to use the optimal number of passes *up to constant factors*. The results are specified for small $\varepsilon > 0$ and integers $k \geq 1$.

matching in the *insertion-only* streaming model. Furthermore, using techniques inspired by approximate maximum matching, we completely resolve the space complexity for *approximate minimum vertex cover* in *one pass* of the *insertion-deletion* streaming model.

1.2 Background

The maximum matching (MM) problem is one of the most studied problems in the streaming model (e.g. [FKM+04, McG05, EKS09, AG11, GKK12, KMM12, Kap13, GO13, CMS13, CCHM15, AG15, Kon15, AKLY16, CCE+16, EHM16, KT17, Kon18b, Tir18, ABB+19, GKMS19, FHM+20, Ber20, DK20, AR20, ALT21, Kap21, CKP+21, AB21, KN21, Ass22, AJJ+22, AS22, FS22, FMU22, BKSW23, KNS23, AS23b, LSZ+23, AS23a, Ass24, ABR24, KN24]). After two

decades, it is now known that computing an exact maximum matching using semi-streaming space is impossible in $o(\log n / \log\log n)$ passes of the stream [GO13]. It is further known that $\Omega(n^{2-o(1)})$ space is required in $o(\sqrt{\log n})$ passes [CKP+21] (see also [AR20]). Although exact MM is still not completely understood for a large number of passes, approximate MM turns out to be even less understood.

In the insertion-only setting, the optimal approximation factor achievable for MM using semi-streaming space is still unknown for any number of passes of the stream, even when considering a few (one, two, or three) passes. Progress, however, has been made by *designing algorithms* that achieve a certain approximation factor and by *proving space lower bounds* that rule out better than a certain approximation factor in a given number of passes using semi-streaming space, respectively. The goal is thus to close the approximation factor gap between algorithms and lower bounds.

1.2.1 Algorithms

The well-known Greedy algorithm in the traditional RAM model begins with an empty matching M and greedily adds edges of the graph to M as long as both endpoints are not already matched by M. This naturally computes a maximal matching M, which is always at least half the size of a maximum matching in the graph, namely, a (1/2)-approximation (see Fact 2.1 for details). Since the algorithm only needs to process each edge once in any arbitrary order and only ever stores the edges of M using $O(|M| \log n) = O(n \log n)$ space, Greedy constitutes a 1/2-approximate one-pass semi-streaming algorithm for MM in the insertion-only streaming model.

To this day, Greedy is still the best-known one-pass semi-streaming algorithm for approximate MM. Doing better than a (1/2)-approximation, even when considering only bipartite graphs, namely, for computing an approximate maximum bipartite matching (MBM), has been an elusive open problem[3]. Improvements, however, have been obtained when allowing even just a few passes of the stream [KMM12, EHM16, KT17, Kon18b, KN21, FS22, BKSW23, KNS23].

Algorithms that achieve a better than 1/2-approximate MBM in a few passes follow a standard approach used by traditional algorithms such as the Hopcroft-Karp algorithm [HK73]: Compute a maximal matching M and then increase its size using a vertex-disjoint set of augmenting paths $\mathcal{P}$, where each path increases the size of M by one. The challenge in the streaming setting has been to develop semi-streaming algorithms that find a set $\mathcal{P}$ that augments the largest number of edges in M using the fewest number of passes of the stream.

Konrad, Magniez, and Mathieu [KMM12] were the first to give a semi-streaming algorithm that achieves a better than (1/2)-approximation using only two passes of the stream. They present both a randomised and a deterministic strategy and show that either can be used

[3] Doing better even using $O(n^{1.999})$ space is still unsolved with one pass of the stream.

to achieve this. Almost all subsequent two-pass semi-streaming algorithms [EHM16, KT17, Kon18b, KN21] crucially use either of these strategies to obtain improvements by solely computing a GREEDY maximal matching in the first pass and then finding a large set $\mathcal{P}$ in the second pass (using one of the strategies). The deterministic strategy has led to a $(7/12 \approx 0.583)$-approximation in two passes using $O(n \log n)$ space [EHM16] and the randomised strategy has led to the current best $(2 - \sqrt{2} \approx 0.585)$-approximation in two passes using $O(n \log n)$ space [Kon18b]. Recently, [BKSW23] gave a generalised deterministic strategy, and thus improving it, which obtains a corresponding algorithm that achieves a $(2 - \sqrt{2} - \varepsilon)$-approximation in two passes using $O(n \log n/\varepsilon)$ space.

Remark 1.1. Certain two-pass semi-streaming algorithms that solely compute GREEDY in the first pass and find augmenting paths in the second pass [KT17, EHM16, BKSW23] have recently been shown to be very successful approaches in designing dynamic algorithms for MBM size estimation in the classic RAM model [BKSW23, Beh23]. This has successfully managed to break the long-standing open question on maintaining a better than $(1/2)$-approximate matching with polylogarithmic update time for the size estimation version of the problem [OR10]. In particular, [BKSW23] obtains a $(2 - \sqrt{2} - \varepsilon)$-approximation for MBM size estimation in $O(\text{poly}(\log n, 1/\varepsilon))$ worst-case update time against adaptive adversaries. See [BKSW23, Beh23, ABR24] for further details.

As a contribution of this thesis and joint work with Konrad [KN21] (see Chapter 5), we obtain a meta-strategy by combining the randomised and deterministic strategies of [KMM12]. We obtain a meta-algorithm (Theorem 3) that yields the two-pass algorithms in [Kon18b] and [EHM16] as special cases. Interestingly, we also obtain a new algorithm that combines both strategies and achieves a $(2-\sqrt{2})$-approximation in two passes using $O(n \log n)$ space, matching the current best algorithm [Kon18b].

In three passes of the stream, a simple deterministic algorithm that does better than a $(1/2)$-approximation was implicitly given in [FKM+04], explicitly stated in [KMM12], and later analysed in [KT17]. This algorithm obtains a $(3/5 = 0.6)$-approximation using $O(n \log n)$ space by computing a GREEDY maximal matching in the first pass and then, in each of the next two passes, by solely running GREEDY on a deterministically specified substream to find a large set $\mathcal{P}$ of vertex-disjoint augmenting paths. Subsequent improvements [EHM16, Kon18b] were naturally obtained from the improved two-pass algorithms as they are essentially one-pass algorithms that find a large enough set $\mathcal{P}$ given a maximal matching M. In particular, since the matching obtained by augmenting M with $\mathcal{P}$ is still a maximal matching, the same algorithm can be used in a third pass to increase its size further. This led to a (≈ 0.606)-approximation using $O(n \log n)$ space [Kon18b]. This trend, however, was recently broken by taking different and more successful approaches tailored for three passes [FS22, KNS23].

As a contribution of this thesis and joint work with Konrad and Steward [KNS23] (see

Chapter 4), we give the current best three-pass algorithm that achieves a $(5/8 = 0.625)$-approximation using $O(n \log n)$ space (Theorem 1), improving on the previous best $(11/18 \approx 0.611)$-approximation using $O(n \log n)$ space [FS22]. Interestingly, our algorithm is once again a simple deterministic algorithm that computes a Greedy maximal matching in the first pass and then, in each of the next two passes, solely runs Greedy on a deterministically specified substream to find a large set $\mathcal{P}$ of vertex-disjoint augmenting paths.

General Graphs. Several of the two- and three-pass algorithms for approximate MBM have also been extended to general graphs, that is, for approximate MM, where similar progress has been made (e.g. [KMM12, KT17, FS22, ABR24]). Typically, the analysis techniques used for MBM give worse approximation guarantees when used for MM due to the additional combinatorial constraints of general graphs (e.g. odd cycles). For example, in three passes for MM, the best-known algorithm achieves a $(82/144 \approx 0.569)$-approximation using $O(n \log n)$ space, which is worse than the $(11/18 \approx 0.611)$-approximation it achieves for MBM [FS22]. Very recently, however, the two-pass semi-streaming algorithm for MBM in [BKSW23] was shown to achieve the same approximation factor for MM [ABR24], thus obtaining a $(2-\sqrt{2}-\varepsilon)$-approximation in two passes using $O(n \log n/\varepsilon^3)$ space[4] – this work also extends the results in Remark 1.1 to general graphs. This recent progress suggests that current techniques for MBM may generally translate to MM without a loss in approximation guarantees, at the very least in two passes of the stream.

Many Passes. Getting better than a $(1/2)$-approximation for MBM using many passes of the stream, where multiplicative constants in the number of passes are typically not important, has been known since the introduction of the semi-streaming model [FKM+04]. Algorithms in this area of study (e.g. [FKM+04, AG11, EKMS12, KT17, ALT21]) have ultimately led to a $(1-\varepsilon)$-approximation in $O(1/\varepsilon^2)$ passes using $O(n \log n)$ space [ALT21]. For MM (e.g. [McG05, AG11, Tir18, FMU22]), a $(1-\varepsilon)$-approximation is achieved in $O(\text{poly}(1/\varepsilon))$ passes using $\tilde{O}(n \,\text{poly}(1/\varepsilon))$ space [FMU22]. Both of these results imply $O(1)$-pass semi-streaming algorithms for any constant $\varepsilon > 0$. We note that a similar, but disjoint, strand of research (e.g. [AG11, AG15, AJJ+22, Ass24]) has culminated in a $(1-\varepsilon)$-approximation in $O(\log n/\varepsilon)$ passes using $O(n \log n/\varepsilon)$ space for both MBM and MM [Ass24]. See [Ass24] for a recent summary of results in this area.

Insertion-Deletion Streams. We highlight here that many of these insertion-only streaming algorithms [FKM+04, McG05, AG11, EKMS12, KMM12, EHM16, KT17, Kon18b, Tir18, ALT21, KN21, AJJ+22, FMU22, FS22, BKSW23, KNS23, ABR24] rely on computing a maximal matching using Greedy as a subroutine. In insertion-deletions streams, however, an adversarially constructed stream can simply delete all the matching edges stored by Greedy

[4] We note here that this algorithm finds paths $\mathcal{P}$ in one pass given a maximal matching and thus can be used to obtain a three-pass algorithm, which should obtain a similar approximation guarantee as [Kon18b]'s three-pass algorithm.

at the end of the stream, which ultimately returns an empty matching. Despite this, Greedy is still a powerful subroutine in the insertion-deletion setting. In particular, it has been used to obtain a one-pass algorithm for α-approximate MM (and MBM) using $O(n^2 \cdot \alpha^3)$ space [AS22], which is optimal (up to constant factors) for the entire range of α [DK20].

As a student-only contribution of this thesis and joint work with Shah [NS22] (see Chapter 7), we show that Greedy is also useful for obtaining a one-pass algorithm for α-approximate minimum vertex cover (MVC) using $O(n^2/\alpha^2)$ space[5], which is optimal (up to constant factors) for the entire range of α [DK20]. We provide a more relevant background on this type of space optimality result for both insertion-only and insertion-deletion streams in Section 1.2.3.

1.2.2 Lower Bounds

Although there has been no improvement over the folklore deterministic Greedy maximal matching algorithm that achieves a (1/2)-approximation in one-pass using semi-streaming space, even when allowed randomisation and restricted to bipartite graphs (approximate MBM), there has been progress on ruling out approximation factors achievable in one pass and, more recently, two or more passes.

Goel, Kapralov, and Khanna [GKK12] were the first to prove a space lower bound for one-pass randomised algorithms for approximate MBM. They show that *Ruzsa-Szemerédi* (RS) graphs [RS78] with linear sized matchings, which are graphs whose edge set is made up of edge-disjoint induced matchings of size $\Theta(n)$, are crucial for proving such results. Building on these ideas, subsequent improvements [Kap13, Kap21] have led to showing that any (even randomised) one-pass algorithm that achieves a better than $(1/(1+\ln 2) \approx 0.591)$-approximation for MBM requires $n^{1+\Omega(1/\log\log n)}$ space, which is strictly more than semi-streaming space. In fact, almost all known lower bounds for exact and approximate MBM [GKK12, Kap13, AR20, Kap21, KN21, CKP+21, Ass22, AS23a, KN24] use RS graphs[6].

Recall that Greedy is an important subroutine in many multi-pass algorithms for approximate MBM. Coupled with the fact that techniques for proving multi-pass lower bounds for arbitrary algorithms were unknown for many years, this motivated the study of lower bounds for specific classes of algorithms [KK20, KN21, KNS23]. In particular, almost all two- and three-pass algorithms for MBM [EHM16, KT17, Kon18b, KN21, BKSW23, KNS23] fall into at least one of the following simple classes of algorithms:

- **Deterministic Greedy-only**: the class of deterministic algorithms that, in each pass, solely run Greedy on a substream of edges [KT17, KNS23];

[5] By convention, we use $\alpha > 1$ for minimisation problems and $\alpha < 1$ for maximisation problems.

[6] The only exceptions we are aware of are [GO13] for exact MBM and [KK20, KNS23] for specific classes of algorithms for approximate MBM.

- **Greedy-first**: the class of (even randomised) algorithms that solely run Greedy on the entire stream of edges in the first pass [EHM16, KT17, Kon18b, KN21, FS22, BKSW23, KNS23].

Studying lower bounds for deterministic Greedy-only algorithms was initiated in [KK20] by considering an even simpler subset of this class. As a contribution of this thesis and joint work with Konrad and Steward [KNS23] (see Chapter 4), we consider the entire class (Theorem 2). In both cases, at least three passes are required to achieve a better than $(1/2)$-approximation. More interestingly, [KK20] show that the $(3/5 = 0.6)$-approximate three-pass algorithm in [KT17] (see also [FKM+04, KMM12]) is optimal in their considered subclass and we show that our $(5/8 = 0.625)$-approximate three pass algorithm (Theorem 1) is optimal in the entire class of Greedy-only algorithms, which is the current best three-pass semi-streaming algorithm. These results thus highlight the strengths and limitations of using only very simple deterministic strategies in each pass.

Remark. Although the simple *randomised* Greedy-only class of algorithms has yet to be systematically studied and thus no lower bounds exist, it is already known that two passes is sufficient to do better than a $(1/2)$-approximation and, indeed, the current best two-pass semi-streaming algorithm that achieves a $(2-\sqrt{2} \approx 0.585)$-approximation [Kon18b] is a randomised Greedy-only algorithm.

Despite the strength of deterministic Greedy-only algorithms in three passes, in two passes, they are essentially no better than the one-pass Greedy algorithm. However, randomised Greedy-only and, more broadly, Greedy-first algorithms have proven to be much more effective in two passes. In fact, almost all known two-pass algorithms [EHM16, KT17, Kon18b, KN21, BKSW23] are Greedy-first algorithms, which includes the current best $(2-\sqrt{2})$-approximate two-pass algorithms in [Kon18b] and our work [KN21] (Theorem 3).

As a contribution of this thesis and joint work with Konrad [KN21] (see Chapter 5), we give the first lower bound for Greedy-first algorithms and show that achieving a better than $(2/3 = 0.667)$-approximation in two passes requires $n^{1+\Omega(1/\log\log n)}$ space (Theorem 4). This naturally implies a limit on the size of $\mathcal{P}$ that is achievable by a one-pass semi-streaming algorithm given a maximal matching, which is the primary algorithmic technique for obtaining two-pass semi-streaming algorithms. As highlighted in Remark 1.1, certain two-pass Greedy-first algorithms have been successfully used to obtain dynamic algorithms in the classic RAM model and thus this lower bound on two-pass Greedy-first algorithms also implies a limitation on such strategies for dynamic algorithms.

Although lower bounds for classes of algorithms have provided a deeper understanding of well-established algorithmic techniques, none of these results apply to arbitrary multi-pass algorithms nor do they give hints to proving such lower bounds, even in just two passes. That

said, after almost two decades since the initial study of multi-pass semi-streaming algorithms for approximate MBM [FKM+04], recent works have successfully developed techniques for proving arbitrary lower bounds [Ass22, AS23a, KN24].

Assadi [Ass22] gave the first lower bound for arbitrary two-pass semi-streaming algorithms, which was subsequently extended to many passes in [AS23a]. These results are obtained by building on the ideas for multi-pass lower bounds for exact MBM [AR20, CKP+21]. However, the qualities of these results are conditioned on the density of RS graphs with linear-sized matchings. Using known RS graph constructions with $n^{\Omega(1/\log\log n)}$ many linear-sized matchings, neither result rules out any constant factor approximations, even in two passes. Assuming that RS graphs with $n^{\Omega(1)}$ many linear-sized matchings exist, both results rule out small constant factor approximations in two [Ass22] or many [AS23a] passes. In the best-case scenario where RS graphs with almost linear many linear sized matchings exist, even [Ass22] only rules out better than (≈ 0.98)-approximations in two passes using semi-streaming space.

As the main contribution of this thesis and joint work with Konrad [KN24] (see Chapter 6), we very recently showed that arbitrary two-pass lower bounds can be obtained using known RS graph constructions, that is, we give the first *unconditional* arbitrary two-pass lower bound for semi-streaming approximate MBM. Using a novel application of RS graphs, we obtain a hierarchical graph construction that appropriately generalises the hard graph construction in [GKK12]. With that, we prove that achieving a better than ($8/9 \approx 0.889$)-approximation in two passes requires $n^{1+\Omega(1/(\log\log n)^2)}$ space, which is strictly more than semi-streaming space (Theorem 6). Our hierarchical technique (applied using a generalisation of RS graphs and with a new non-standard round-elimination argument) has since been used to obtain an optimal (up to constant factors) multi-pass semi-streaming lower bound for the maximal independent set problem [AKNS24].

We emphasise here that all the lower bounds that we have discussed are for approximate MBM and thus imply the same results for approximate MM. They also imply the same results in the insertion-deletion setting; however, significantly stronger lower bounds are already known for approximate MBM in one pass of the stream [DK20] (see also [AKLY16]).

1.2.3 Space Optimality

In this subsection, as a brief aside from approximate MBM in a few passes[7], we discuss streaming algorithms that use optimal space up to constant factors, the recent interest in their study, and where Greedy has been a useful subroutine.

Observe first that any α-approximate MBM has $\Omega(n \cdot \alpha)$ many edges in a graph that has a maximum matching of size $\Theta(n)$. Furthermore, each edge needs $\Omega(\log n)$ bits of space to store

[7]It is possible (and even suggested) to read this part of the thesis after an initial pass of the content that relates to approximate MBM in a few passes, i.e., initially ignoring Sections 1.2.3, 1.3.4 and 3.4, and Chapter 7.

in memory, for example, an identifier for each endpoint. This trivially implies an $\Omega(n \cdot \alpha \cdot \log n)$ space lower bound for any streaming algorithm that outputs an α-approximate MBM since simply returning the solution requires this amount of space. Therefore, many of the insertion-only algorithms that we have discussed so far, which obtain constant factor approximations for MBM (e.g. [FKM+04, EKMS12, EHM16, KT17, Kon18b, ALT21, KN21, AJJ+22, FS22, KNS23]), use the optimal $O(n \log n)$ space up to constant factors. This, however, has not been as obvious for other graph problems, even in the insertion-only model.

In their seminal work [FKM+05], Feigenbaum at al. show that, for testing the connectivity or bipartiteness of a graph, there is a one-pass algorithm for insertion-only streams that stores $O(n)$ edges (of a spanning tree) and thus uses $O(n \log n)$ bits of space. Unlike algorithms for approximate MBM, an algorithm for testing connectivity or bipartiteness only needs to output a single bit and thus there is no trivial space lower bound based on the size of the solution. To that end, Feigenbaum et al. also show that any such one-pass algorithm requires $\Omega(n)$ bits of space, thus proving space optimality of their algorithm *up to logarithmic factors*.

For a very long time, space optimality up to logarithmic factors was considered tight enough for graph streaming problems. However, such a gap leaves open the question of whether the logarithmic factors are necessary or whether other algorithmic techniques can be used to remove them. In the case of connectivity, the logarithmic gap left open the question of whether storing the $O(n)$ edges of a spanning tree could be improved, for example, by storing some sketch of $O(n/\log n)$ edges instead. After about a decade, the space lower bound was improved to $\Omega(n \log n)$ [SW15], showing that the logarithmic factor is indeed required.

Similar progress has also been made in the insertion-deletion model. In their seminal work, Ahn, Guha, and McGregor [AGM12a] gave a one-pass algorithm for computing a spanning forest using $O(n \log^3 n)$ bits of space. Their algorithm relies on randomly sampling edges using ℓ_0-samplers, which optimally uses $\Theta(\log^3 n)$ bits of space for each sampled edge[8] [JST11, KNP+17]. This is a standard algorithmic technique for insertion-deletion algorithms (e.g. [AGM12a, AGM12b, Kon15, CCE+16, AKLY16, Kon21, KK22, AS22, NS22]) since deterministically outputting an edge of the input graph essentially requires storing the entire graph. The question of whether the logarithmic factors due to the ℓ_0 samplers are inherently required was left open for several years until a space lower bound of $\Omega(n \log^3 n)$ bits [NY19] proved its necessity.

A perhaps more interesting result is the recent progress on one-pass α-approximate MM in insertion-deletion streams. The space lower bound of $\Omega(n^2 \cdot \alpha^3)$ bits [DK20] (see also [AKLY16]), the algorithms in [AKLY16, CCE+16] that use ℓ_0-samplers and $O(n^2 \cdot \alpha^3 \cdot \log^4 n)$ bits of space, and the above results for computing a spanning forest seem to indicate that the logarithmic factors should be required. Surprisingly, an improved algorithm using $O(n^2 \cdot \alpha^3)$ bits

[8] This optimal space bound applies when the probability of success is at least $1 - \frac{1}{\text{poly}(n)}$.

of space [AS22] showed that different algorithmic techniques are able to remove the logarithmic factors. In fact, their algorithm uses Greedy as a subroutine, which was not used by previous algorithms [AKLY16, CCE+16].

As previously mentioned, in a student-only contribution of this thesis and joint work with Shah [NS22] (see Chapter 7), we obtain a similar result for one-pass α-approximate MVC in insertion-deletion streams – where we use the convention that $\alpha > 1$ for minimisation problems. The previous algorithm in [DK20] uses counters to test whether edges are present in $O(n^2/\alpha^2)$ many edge-disjoint subgraphs that may have up to $O(\alpha^2)$ edges each[9]. This uses $O(\log \alpha)$ bits per counter and overall their algorithm uses $O((n^2/\alpha^2) \cdot \log \alpha)$ bits of space, which matches the lower bound in [DK20] up to the logarithmic factor. Using Greedy as a subroutine, we remove the logarithmic factor and obtain an algorithm that uses $O(n^2/\alpha^2)$ bits of space, which is optimal up to constant factors (Theorem 8).

1.3 Our Contributions

All contributions of this thesis have been peer-reviewed and published. In joint works, the authors' names are in alphabetical order, and the author of this thesis has been a driving force throughout the collaborations.

The following works, listed in chronological order, are contributions of this thesis:

- [KN21] in Chapter 5: Christian Konrad and Kheeran K. Naidu. On Two-Pass Streaming Algorithms for Maximum Bipartite Matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021.*
- [NS22] in Chapter 7 (**student-only**): Kheeran K. Naidu and Vihan Shah. Space Optimal Vertex Cover in Dynamic Streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022.*
- [KNS23] in Chapter 4: Christian Konrad, Kheeran K. Naidu, and Arun Steward. Maximum Matching via Maximal Matching Queries. In *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023.*
- [KN24] in Chapter 6: Christian Konrad and Kheeran K. Naidu. An Unconditional Lower Bound for Two-Pass Streaming Algorithms for Maximum Matching Approximation. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024.*

In this thesis, we study the approximate maximum bipartite matching (MBM) problem in the insertion-only streaming model. We make contributions towards understanding the optimal approximation factor achievable using semi-streaming space in only a few (one, two, and three)

[9] Using counters is another algorithmic technique used in insertion-deletion streams when only learning the existence of an edge in a subgraph is required (e.g. [CCE+16]) instead of recovering one (using an ℓ_0 sampler).

passes of the stream. In particular, we give algorithms and lower bounds for Greedy-only, Greedy-first, and arbitrary algorithms. Furthermore, we study the approximate minimum vertex cover (MVC) problem in the insertion-deletion streaming model. In particular, we obtain a one-pass algorithm the uses the optimal space (up to constant factors) for the full range of approximation factors.

1.3.1 Greedy-only Approximate MBM

As our first set of contributions and joint work with Konrad and Steward [KNS23], we consider the class of deterministic Greedy-only algorithms and obtain a complete understanding of them in one, two, and three passes in the insertion-only model. This class of algorithms is probably the most straightforward and most natural approach for approximate MBM, which includes the very first $(3/5 = 0.6)$-approximate three-pass algorithm that uses optimal $O(n \log n)$ space [KT17] (see [FKM+04, KMM12]). See Table 1.2 for a summary of our results in context of the literature.

Approx.	Space	Type	Ref.	Comments
One-Pass Greedy-only MBM				
$1/2$	$O(n \log n)$	det. alg	folklore	Greedy
$1/2$	any	det. LB	**our work**	# of passes LB
Two-Pass Greedy-only MBM				
$1/2$	$O(n \log n)$	det. alg	folklore	Greedy (in one pass)
$2 - \sqrt{2} \approx 0.585$	$O(n \log n)$	rand. alg	[Kon18b]	implies a 3-pass alg
$1/2$	any	det. LB	**our work**	# of passes LB
Three-Pass Greedy-only MBM				
$3/5 = 0.6$	$O(n \log n)$	det. alg	[KT17]	see [FKM+04, KMM12]
≈ 0.606	$O(n \log n)$	rand. alg	[Kon18b]	–
$5/8 = 0.625$	$O(n \log n)$	det. alg	**our work**	–
$5/8 = 0.625$	any	det. LB	**our work**	# of passes LB

Table 1.2: A summary of our deterministic Greedy-only MBM results in the context of the relevant literature, including randomised Greedy-only MBM. Results in each category are presented in chronological order. The unshaded rows correspond to algorithms while the shaded (grey) ones correspond to lower bounds. Randomised and deterministic are abbreviated as rand. and det., respectively. Algorithm and lower bound are abbreviated as alg and LB, respectively.

In each pass, a Greedy-only algorithm solely computes a Greedy maximal matching on a substream of edges, corresponding to a subgraph of the input graph $G = (A, B, E)$. To

systematically study these algorithms, we abstract the computation of the GREEDY maximal matching in each pass. We thus represent each pass as an edge-query $Q \subseteq A \times B$ to an oracle that returns a maximal matching M in the corresponding edge-induced subgraph $G[Q \cap E]$. As in the streaming algorithm, the query may be based on the responses to the previous queries, i.e., the query is adaptive. Therefore, the results we obtain in $r \geq 1$ rounds of the query model apply to the class of GREEDY-only algorithms in r passes of the stream – see Chapter 2 for exact details of this relationship.

Query Model Results

Theorem 1. *There is an adaptive deterministic edge-query algorithm for MBM that achieves an* $(5/8)$*-approximation in three query rounds.*

Theorem 2. *There does* ***not*** *exist an adaptive deterministic edge-query algorithm for MBM that achieves a better than*

1. $(1/2)$*-approximation in one query round,*
2. $(1/2 + o(1))$*-approximation in two query rounds, or*
3. $(5/8 + o(1))$*-approximation in three query rounds.*

By additionally considering the trivial one-round algorithm that queries the entire input graph and obtains a (1/2)-approximate maximal matching, which also applies to two rounds[10], our results are optimal in one, two, and three query rounds. In particular, doing better than a (1/2)-approximation requires three rounds, where our $(5/8 = 0.625)$-approximation algorithm is optimal.

Discussion & Semi-Streaming Implications

As intended, our results in the considered query model have direct implications on the class of deterministic GREEDY-only streaming algorithms in insertion-only streams.

Algorithms in this query model immediately translate into GREEDY-only streaming algorithms where each adaptive edge-query determines the substream of edges for which GREEDY computes a maximal matching in each pass. In general, the space the streaming algorithm uses depends on the number of query rounds and the space required to represent each query. With only a constant number of query rounds and simple to represent queries in each round, our three-round query algorithm (Theorem 1) constitutes an (5/8)-approximate three-pass

[10] In two rounds, after obtaining a maximal matching M in the first round, the additional second round can be used to query all edges that are incident to an arbitrary edge $e \in M$ and have one endpoint unmatched by M. This either finds edges to augment e or determines that e is not part of a length-3 augmenting path. Either way, the approximation factor is improved to $1/2 + o(1)$.

streaming algorithm that uses the optimal $O(n \log n)$ space. Our Greedy-only algorithm thus improves on the previous best $(11/18 \approx 0.611)$-approximate three-pass semi-streaming algorithm [FS22].

On the other hand, lower bounds on the approximation factor achievable in r rounds implies the same lower bound for r-pass Greedy-only streaming algorithms by making the oracle *streaming consistent*[11] (see Chapter 2 for details). Therefore, our results (Theorem 2) imply that deterministic Greedy-only streaming algorithms require at least three passes to achieve a better than $(1/2)$-approximation. Furthermore, in three passes, our $(5/8)$-approximation algorithm is best possible. Note that this holds even for deterministic Greedy-only streaming algorithms that use $O(n^2)$ space, not just semi-streaming space, since our results do not restrict the space required to represent the queries.

1.3.2 Greedy-first Approximate MBM

As our next set of contributions and joint work with Konrad [KN21], we study the class of Greedy-first semi-streaming algorithms for approximate MBM and substantially narrow the approximation factor gap for two-pass algorithms. This class of algorithms includes almost all known two-pass semi-streaming algorithms [EHM16, KT17, Kon18b, KN21, BKSW23] and the best-known two-pass algorithm that achieves a $(2 - \sqrt{2} \approx 0.585)$-approximation using optimal $O(n \log n)$ space [Kon18b]. See Table 1.3 for a summary of our results in context of the literature.

A Greedy-first algorithm solely computes a Greedy maximal matching in the first pass and has any arbitrary (even randomised) subsequent passes, if any. Hence, the $(1/2)$-approximate Greedy maximal matching algorithm is the only one-pass Greedy-first algorithm and is thus trivially optimal. The class of algorithms becomes more exciting in two passes, where we obtain a new algorithm and the first lower bound.

Two-Pass Algorithm

Theorem 3. *For $p \in (0, 1]$ and integer $d \geq 1$, there is a two-pass streaming algorithm for MBM with approximation factor*

$$\begin{cases} \frac{1}{2} + (\frac{1}{d+p} - \frac{1}{2d}) \cdot p, & \text{if } p \leq d(\sqrt{2} - 1) \\ \frac{1}{2} + \frac{d-p}{6d+2p}, & \text{otherwise,} \end{cases}$$

(ignoring lower order terms) that succeeds with probability at least $1 - 1/\operatorname{poly}(n)$ and uses $O(n \log n)$ bits of space.

[11] A similar definition was first seen in [KK20] for vertex-query maximal matching oracles.

Approx.	Space	Type	Ref.	Comments
One-Pass Greedy-first MBM				
1/2	$O(n \log n)$	det. alg	folklore	GREEDY
1/2	–	rand. LB	trivial	only alg is GREEDY
Two-Pass Greedy-first MBM				
$9/16 \approx 0.562$	$O(n \log n)$	det. alg	[KT17]	–
$7/12 \approx 0.583$	$O(n \log n)$	det. alg	[EHM16]	implies a 3-pass alg
$2 - \sqrt{2} \approx 0.585$	$O(n \log n)$	rand. alg	[Kon18b]	implies a 3-pass alg
$2 - \sqrt{2} \approx 0.585$	$O(n \log n)$	rand. alg	**our work**	–
$2/3 \approx 0.667$	$n^{1+\Omega(1/\log\log n)}$	rand. LB	**our work**	–
$2 - \sqrt{2} - \varepsilon$	$O(n \log n/\varepsilon)$	det. alg	[BKSW23]	–
Three-Pass Greedy-first MBM				
$3/5 = 0.6$	$O(n \log n)$	det. alg	[KT17]	see [FKM+04, KMM12]
≈ 0.605	$O(n \log n)$	det. alg	[EHM16]	–
≈ 0.606	$O(n \log n)$	rand. alg	[Kon18b]	–
$11/18 = 0.611$	$O(n \log n)$	det. alg	[FS22]	–
$5/8 = 0.625$	$O(n \log n)$	det. alg	**our work**	–

Table 1.3: A summary of our GREEDY-first MBM results in the context of the relevant literature. Results in each category are presented in chronological order. The unshaded rows correspond to algorithms while the shaded (grey) ones correspond to lower bounds. Where applicable, the results are specified for small $\varepsilon > 0$. Randomised and deterministic are abbreviated as rand. and det., respectively. Algorithm and lower bound are abbreviated as alg and LB, respectively.

Our meta-algorithm is a GREEDY-first algorithm and yields the algorithms in [EHM16, KT17] and [Kon18b] as special cases, thereby unifying two strands of research. Two parameter settings optimise the approximation factor of our algorithm (see Figure 1.4) and obtain a $(2 - \sqrt{2} \approx 0.585)$-approximation in two passes using optimal $O(n \log n)$ space. In particular, by setting $p = \sqrt{2} - 1$ and $d = 1$, we obtain the two-pass algorithm in [Kon18b], and more interestingly, by setting $p = 2\sqrt{2} - 2$ and $d = 2$, we obtain a new algorithm. Additionally, by setting $p = 1$ and $d = 3$, we obtain the $(7/12 \approx 0.583)$-approximation two-pass algorithm in [EHM16]. We prove that the analysis of our meta-algorithm is tight for all interesting parameter settings, namely, $p \leq d(\sqrt{2} - 1)$, which includes our new algorithm and the algorithms in [EHM16, Kon18b].

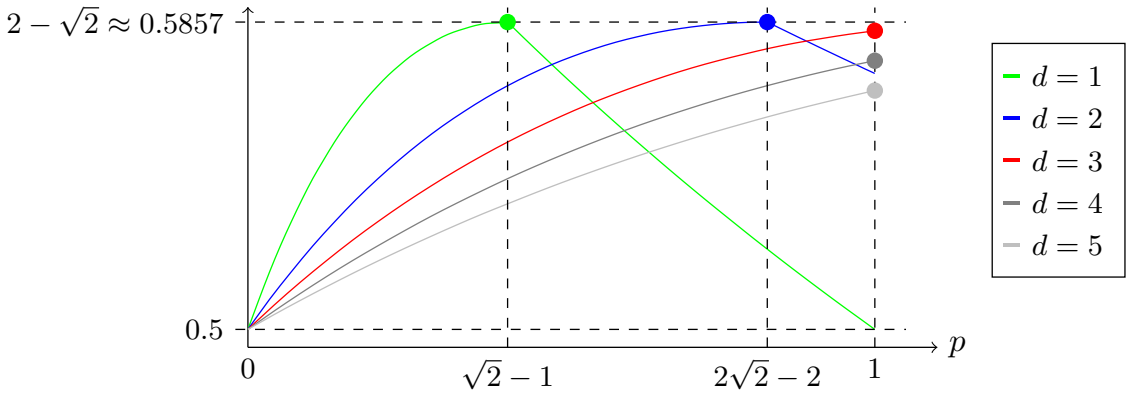


Figure 1.4: A plot of the approximation factors of our two-pass meta-algorithm for different settings of d. For $d \geq 3$, the guarantees progressively get worse and so we omit $d \geq 6$ from the plot. The optimal approximation factor of $2-\sqrt{2}$ is achieved when $d = 1, 2$ and $p = d\cdot(\sqrt{2}-1)$.

Two-Pass Lower Bound

Theorem 4. *For constant $\epsilon > 0$, any constant-error (possibly randomised) Greedy-first streaming algorithm for MBM that achieves a $(2/3+\epsilon)$-approximation in two passes requires $n^{1+\Omega(1/\log\log n)}$ bits of space.*

Our result is the first lower bound for Greedy-first algorithms and rules out better than $(2/3)$-approximations using semi-streaming space since $n^{\Omega(1/\log\log n)} \gg \operatorname{polylog} n$. To obtain our result, we prove the existence of the following RS graph construction, which may be of independent interest.

Theorem 5. *For any small enough constant $\delta > 0$ and sufficiently large integer $n \geq 1$, there exists a $(2n)$-vertex balanced bipartite (r,t)-RS graphs with $r = (\frac{1}{2}-\delta)\cdot n$ and $t = n^{\Omega(1/\log\log n)}$ such that there are $t/2$ edge-disjoint near-perfect matchings of size $2r = (1-2\delta)\cdot n$.*

Discussion

Our meta-algorithm (Theorem 3) shows that the deterministic and randomised strategies in [KMM12], even combined, are currently not able to achieve a better than $(2-\sqrt{2}\approx 0.585)$-approximation for two-pass Greedy-first semi-streaming algorithms, suggesting that other algorithmic strategies, if any, are needed. Furthermore, our lower bound (Theorem 4) shows that such algorithms, even ones that use other algorithmic strategies, are not able to achieve a better than $(2/3 \approx 0.667)$-approximation. For example, it implies a bound on the number of vertex-disjoint augmenting paths that any one-pass semi-streaming algorithm finds given a maximal matching, which is the primary algorithmic technique for two-pass algorithms [EHM16, KT17, Kon18b, KN21, BKSW23].

We obtain our lower bound result by building on the (2/3)-approximation lower bound for one-pass semi-streaming algorithms for MBM in [GKK12]. As pointed out by Kapralov [Kap21], their techniques for obtaining the current best $((1+\ln 2)^{-1})$-approximation lower bound could possibily be applied to a construction by Huang et al. [HPT+19], which would obtain a $(2-\sqrt{2})$-approximation lower bound for one-pass semi-streaming algorithms. It is unclear whether a two-pass lower bound for Greedy-first semi-streaming algorithms could be obtained from this. However, if possible, it would show that a $(2-\sqrt{2})$-approximation is optimal within the class of two-pass Greedy-first algorithms.

Finally, recall from Remark 1.1 that recent work has translated certain two-pass Greedy-first algorithms for MBM into dynamic algorithms for MBM size-estimation in the classic RAM model [BKSW23, Beh23]. Therefore, our lower bound on two-pass Greedy-first algorithms (Theorem 4) also implies limitations on the corresponding dynamic algorithms. In particular, our result rules out better than (2/3)-approximations for MBM size estimation for these type of dynamic algorithms, i.e., other techniques are required for further improvements.

1.3.3 Arbitrary Approximate MBM

As our main contribution and joint work with Konrad [KN24], we study arbitrary semi-streaming algorithms and make progress in narrowing the approximation gap in two passes. See Table 1.5 for a summary of our results in context of the literature.

The best-known one-pass semi-streaming algorithm is the trivial (1/2)-approximate Greedy algorithm, and the best-known one-pass lower bound rules out better than $((1+\ln 2)^{-1} \approx 0.591)$-approximations using semi-streaming space [Kap21]. In two passes and using semi-streaming space, the best-known algorithms in [Kon18b] and our work (Theorem 3) achieve a $(2-\sqrt{2} \approx 0.585)$-approximation and, before our work, no lower bounds unconditionally ruled out any constant factor approximations, not even a (0.999)-approximation. Our work is thus the first *unconditional* two-pass lower bound for *arbitrary* semi-streaming algorithms.

Theorem 6. *For constant $\epsilon > 0$, any constant-error (possibly randomised) two-pass streaming algorithm for MBM that achieves a $(8/9+\epsilon)$-approximation requires $n^{1+\Omega(1/(\log\log n)^2)}$ bits of space.*

We obtain the hard graph construction used to prove our result using a novel application of RS graphs to 'hide' *many* independent instances of the hard one-pass bipartite graph construction in [GKK12] (see Section 3.3 for details). Our construction uses known RS graphs with linear (in the number of vertices n) sized matchings; therefore, it is an unconditional result. In contrast, the previous lower bound result by Assadi [Ass22] uses different techniques and

Approx.	Space	Type	Ref.	Comments
One-Pass Arbitrary MBM				
$1/2 = 0.5$	$O(n \log n)$	det. alg	folklore	GREEDY
$2/3 \approx 0.667$	$n^{1+\Omega(1/\log\log n)}$	rand. LB	[GKK12]	–
$1 - \frac{1}{e} \approx 0.633$	$n^{1+\Omega(1/\log\log n)}$	rand. LB	[Kap13]	–
$(1 + \ln 2)^{-1} \approx 0.591$	$n^{1+\Omega(1/\log\log n)}$	rand. LB	[Kap21]	–
Two-Pass Arbitrary MBM				
$519/1000 = 0.519$	$O(n \log n)$	algs	[KMM12]	rand./det. algs
$9/16 \approx 0.562$	$O(n \log n)$	det. alg	[KT17]	–
$7/12 \approx 0.583$	$O(n \log n)$	det. alg	[EHM16]	implies a 3-pass alg
$2 - \sqrt{2} \approx 0.585$	$O(n \log n)$	rand. alg	[Kon18b]	implies a 3-pass alg
$2 - \sqrt{2} \approx 0.585$	$O(n \log n)$	rand. alg	**our work**	–
$[0.98, 1 - o(1)]$	$\omega(n \operatorname{polylog} n)$	rand. LB	[Ass22]	conditional LB
$2 - \sqrt{2} - \varepsilon$	$O(n \log n/\varepsilon)$	det. alg	[BKSW23]	–
$8/9 \approx 0.889$	$n^{1+\Omega(1/(\log\log n)^2)}$	rand. LB	**our work**	–
Three-Pass Arbitrary MBM				
$3/5 = 0.6$	$O(n \log n)$	det. alg	[KT17]	see [FKM+04, KMM12]
≈ 0.605	$O(n \log n)$	det. alg	[EHM16]	–
≈ 0.606	$O(n \log n)$	rand. alg	[Kon18b]	–
$11/18 = 0.611$	$O(n \log n)$	det. alg	[FS22]	–
$5/8 = 0.625$	$O(n \log n)$	det. alg	**our work**	–

Table 1.5: A summary of our arbitrary MBM results in the context of the relevant literature. Results in each category are presented in chronological order. The unshaded rows correspond to algorithms while the shaded (grey) ones correspond to lower bounds. Where applicable, the results are specified for small $\varepsilon > 0$. Randomised and deterministic are abbreviated as rand. and det., respectively. Algorithm and lower bound are abbreviated as alg and LB, respectively.

relies on assumptions about the density of such RS graphs in their hard graph construction[12]. Under the best-case assumption that RS graphs with near-linear many linear-sized matchings exist [FHS17], they rule out better than (≈ 0.98)-approximations in two passes using semi-streaming space. However, using known RS graphs with $n^{\Omega(1/\log\log n)}$ many linear-sized matchings [FLN+02] (see also [GKK12]), they do not rule out any constant factor approximations.

We formally prove the hardness of our graph construction by abstracting its hardness as a new communication game HiddenStrings, which can be seen as a generalisation of Index.

[12]Note that progress on understanding the density of such RS graphs has been a long-standing open problem in combinatorics [FLN+02, FHS17]

Lower bounds in the communication model imply lower bounds in the streaming model by the well-known connection between the two models (see Chapter 2 for details). Then, using the information cost tradeoff result for the Index problem in the two-party communication model established in [JRS09][13], we obtain the following result that may be of independent interest.

Theorem 7. *For any constant $\delta < \frac{1}{2}$, any δ-error two-round protocol for* HiddenStrings *requires* $\Omega(\min\{t_g \cdot r_g, r_g \cdot t_\ell \cdot r_\ell\})$ *bits of communication for any integers* $t_g, r_g, t_\ell, r_\ell \geq 1$.

Discussion

Our work presents a novel approach that 'hides' many independent instances of a hard bipartite (or 2-colourable) graph construction using RS graphs. Although our proof technique (using the information-cost tradeoff result for Index) is limited to proving two-pass lower bounds, our 'hiding' approach can be recursively applied to obtain a hard graph construction for MBM in three or more passes. This would obtain a hierarchical family of graphs that would rule out better than $(1-\varepsilon)$-approximations for MBM in $o(\log(1/\varepsilon))$ passes using semi-streaming space, matching a recent result by Assadi and Sundaresan [AS23a], but unconditionally, i.e., without any assumptions on the density of RS graphs with linear-sized matchings.

Recent work by Assadi et al. [AKNS24] has given a different proof technique (round-elimination with message compression) that can be used to prove the aforementioned multi-pass lower bound for MBM. More interestingly, they give a generalisation of RS graphs that extends our approach to hiding many independent instances of a hard k-colourable graph construction (for $k \leq 2^{(\log n)^{1/4}}$). This allows for the construction of hard instances for problems where a bipartite graph is trivially easy. They use this to give a hierarchical family of hard graph constructions to show that any p-pass algorithm for the maximal independent set problem (MIS) requires $\Omega(n^{1+1/(2^p-1)-o(1)})$ space in insertion-only streams, i.e., $\Omega(\log\log n)$ passes are required using semi-streaming space, which matches (up to constant factors) the long-standing $O(\log\log n)$-pass semi-streaming algorithm for MIS [ACG+15] in insertion-deletion streams[14].

1.3.4 Insertion-Deletion Approximate MVC

As a student-only contribution and joint work with Shah [NS22], we study the approximate MVC problem in one-pass in the insertion-deletion streaming model. See Table 1.6 for a summary of our results in context of the literature.

Theorem 8. *There is a one-pass insertion-deletion streaming algorithm for α-approximate MVC that succeeds with probability at least $1 - 1/\operatorname{poly}(n)$ and uses $O(n^2/\alpha^2)$ bits of space for any $\alpha \leq n^{1-\delta}$ where $\delta > 0$.*

[13] To the best of our knowledge, we are the first to exploit this tradeoff in the context of multi-pass lower bounds for graph streaming algorithms.

[14] Prior to this result, multi-pass lower bounds for MIS were not known, even for two passes of insertion-deletion streams.

Approx.	Space	Type	Ref.	Comments
One-Pass Insertion-Deletion MVC				
α	$\Omega(n^2/\alpha^2)$	rand. LB	[DK20]	–
α	$O((n^2/\alpha^2) \cdot \log \alpha)$	det. alg	[DK20]	–
α	$O(n^2/\alpha^2)$	rand. alg	**our work**	uses GREEDY

Table 1.6: A summary of our insertion-deletion MVC results in the context of the relevant literature. Results in each category are presented in chronological order. The unshaded rows correspond to algorithms while the shaded (grey) ones correspond to lower bounds. Randomised and deterministic are abbreviated as rand. and det., respectively. Algorithm and lower bound are abbreviated as alg and LB, respectively.

The space used by our algorithm matches the $\Omega(n^2/\alpha^2)$ space lower bound for algorithms that achieve an α-approximation for MVC up to constant factors [DK20], thus improving on the previous best algorithm that uses $O((n^2/\alpha^2) \cdot \log \alpha)$ bits of space [DK20].

Discussion

Our space optimal algorithm for approximate MVC (Theorem 8) and the space optimal algorithm for approximate MM [AS22] both use GREEDY as a subroutine (see Section 3.4 for a comparison). This relationship between MM and MVC via GREEDY is not uncommon in the streaming model. In insertion-only streams, GREEDY is the best-known one-pass semi-streaming algorithm for approximate MM and approximate MVC, achieving a (1/2)-approximation in both cases. In sliding window streams, where the goal is to solve a graph problem on the L most recent edges of the stream, the best-known algorithm for approximate MM uses GREEDY as a subroutine and achieves a $(3+\varepsilon)$-approximation for $\varepsilon > 0$ [CMS13]. The same algorithm with only minor modifications (for bookkeeping) has also been shown to achieve a $(3+\varepsilon)$-approximation for MVC [Nal21].

A key difference between approximate MVC and approximate MM in insertion-deletion streams, however, is that deterministic algorithms are generally not useful for approximate MM. In particular, an adversarially constructed stream can easily delete the matching edges in the output of a deterministic algorithm. This is not the case for approximate MVC since the output is a subset of vertices (that can not be deleted by an adversary). In fact, the previous best algorithm is a deterministic algorithm that achieves an α-approximation using $O((n^2/\alpha^2) \cdot \log \alpha)$ bits of space [DK20]. Although our algorithm improves on this to match the space lower bound in [DK20], it relies on randomisation and thus only settles our understanding of randomised algorithms. However, for deterministic algorithms, whether it is possible to prove a stronger lower bound or obtain a better algorithm is still an open question.

1.4 Thesis Structure

The remainder of this thesis is structured as follows:

In Chapter 2, we provide some general preliminaries for our work, which will be useful throughout the thesis and especially in Chapter 3. In Chapter 3, we give a detailed overview of the technical ideas needed to prove our results. In Chapters 4 to 7, we present our relevant published papers as chapters where Chapter 4 corresponds to [KNS23], Chapter 5 corresponds to [KN21], Chapter 6 corresponds to [KN24], and Chapter 7 corresponds to [NS22].

Remark. Chapters 4 to 7 are officially *publications as chapters* and are thus self-contained. This means that a reader may read the relevant chapters without reading any other part of this thesis. Similarly, a reader who has been through the relevant preliminaries and technical overview in Chapters 2 and 3 may skip directly to the main technical content of the corresponding chapter (reviewing the specific preliminaries if required).

2 Preliminaries

Basic Notation. Unless stated otherwise, log denotes the binary logarithm, and ln denotes the logarithm to the base e. For any positive integer n, we define the set of n integers $[n] := \{1, 2, \ldots, n\}$. For any pair of real numbers c, d, we define the open-closed interval $(c, d] := \{x \mid c < x \leq d\}$ and similarly its variants: closed-open, open (open-open), and closed (closed-closed). For any set X, we denote the cardinality of X as $|X|$. For any two sets X and Y, we define $X \oplus Y := (X \backslash Y) \cup (Y \backslash X)$ as their symmetric difference. For any n-bit-string $Z \in \{0,1\}^n$ and integer $J \in [n]$, the J^{th} bit of Z is denoted $Z[J]$.

Graph Notation. Let $G = (V, E)$ be a graph where $|V| = n$ and $E \subseteq V \times V$. We say that $H = (V, F)$ is a **subgraph** of G if $F \subseteq E$. For any vertex $v \in V$ and subgraph $H = (V, E)$ of G, we denote $N_H(v) = \{uv \in F \mid u \in V\}$ as the **neighbourhood** of v in H. For any edge $e = uv \in E$, we say that it has the **endpoints** u and v or, additionally, it is **incident** to the vertices u and v. For any set of edges F (not necessarily a subset of E), we define $G[F] := (V, E \cap F)$ to be the subgraph induced by the edges F. For any subset of vertices $U \subseteq V$, we define $G[U] := (V, E \cap (U \times U))$ to be the subgraph induced by the vertices U. Furthermore, we define $U(F) := \{u \in U \mid uv \in F\}$ to be the vertices of U that are also endpoints of an edge in F. The graph G is a **simple graph** when E is not a multi-set and does not contain any self-loops (edges with the same endpoints). The graph G is a **bipartite graph** when $V = A \cup B$ such that $A \cap B = \emptyset$ and $E \subseteq A \times B$. For simplicity, we write $G = (A, B, E)$ to denote a bipartite graph. We call a bipartite graph $G = (A, B, E)$ **balanced** if $|A| = |B|$.

Asymptotic Notation. For any function f, we use the standard $O(f)$, $o(f)$, $\Omega(f)$, $\omega(f)$, $\Theta(f)$ notation. For simplicity of exposition, we sometimes hide logarithmic factors using 'soft' notation where, for example, $\tilde{O}(f) = O(f \operatorname{polylog} f)$.

Approximations. Let P be a problem with an optimal value $\textsc{opt}$ and let $\mathcal{A}$ be an algorithm for problem P that returns an output of value $\textsc{out}$. We say that $\textsc{out}$ is an α-approximation for P as long as $\textsc{out} \geq \alpha \cdot \textsc{opt}$. By convention, if P is a *maximisation* problem, we consider ap-

proximation factors $0 < \alpha < 1$, and if P is a *minimisation* problem, we consider approximation factors $\alpha > 1$.

Maximum Matching

Let $G = (V, E)$ be a n-vertex graph. A **matching** in G is a set of vertex-disjoint edges, and a **maximum matching** is one of maximum size. We denote the size of a maximum matching as $\mu(G)$. A maximum matching is called a **perfect matching** if it matches every vertex, namely, $\mu(G) = n/2$. A **maximal matching** M is a matching that is inclusion-wise maximal – for every edge $e \in E \backslash M$, $M \cup \{e\}$ is not a matching.

Every maximum matching is a maximal matching, but not every maximal matching is maximum. Their sizes are, however, closely related. Any edge of a maximal matching can be incident to at most two edges of a maximum matching, and thus, by a simple counting argument, we have the following fact:

Fact 2.1. For any graph G, every maximal matching M is such that

$$|M| \geq \mu(G)/2,$$

which is a $(1/2)$-approximate maximum matching.

Augmenting Paths. For any matching M of a graph G, we define an **alternating path** as a path that begins with a vertex unmatched by M and alternates between edges in M and not in M. An **augmenting path** P is then defined as an odd-length alternating path, which begins *and* ends with an unmatched vertex. Augmenting a matching M using P obtains a new matching M' by removing the edges of $P \cap M$ from M and replacing them with the edges of $P \backslash M$, which increases its size by one. If M is a maximal matching, then M' is also maximal since all vertices matched in M are still matched in M'. When given a set of vertex-disjoint augmenting paths $\mathcal{P}$, these can simultaneously be used to augment M and increase its size by $|\mathcal{P}|$. In particular, we have the following fact:

Fact 2.2. For any graph G with a maximum matching M^* and matching M, their symmetric difference $M^* \oplus M$ is exactly the set of vertex-disjoint augmenting paths required to augment M into the maximum matching M^*.

Matching theory is a well-studied area of research, and we refer the interested reader to the book by Lovász and Plummer [LP09] for more details.

Ruzsa-Szemerédi (RS) Graphs

A n-vertex (r,t)-RS graph is a graph whose edge set is made up of t many edge-disjoint induced matchings of size r. In particular, it is a graph $G = (V, E)$ where

- $E = M_1 \cup M_2 \cup \cdots \cup M_t$ such that $M_i \cap M_j = \emptyset$ for every $i \neq j \in [t]$; and
- For each $i \in [t]$, the edges M_i form a matching such that $|M_i| = r$ and $G[V(M_i)] = G[M_i]$.

Throughout this thesis, we consider $2n$-vertex balanced bipartite (r,t)-RS graphs that have matchings of size linear in the number of vertices, that is, $r = \Theta(n)$. For (r,t)-RS graphs where $r \geq n/2$, it is known that $t = O(\log n)$ [FHS17] and thus the graphs are not sufficiently dense for our applications. Therefore, we consider (r,t)-RS graphs where r is slightly smaller than $n/2$ and significantly denser graphs are known to exist.

Proposition 2.3 (cf. [GKK12] (see also [FLN+02]))**.** *For any small constant $\delta > 0$ and sufficiently large integer $n \geq 1$, there exists a $(2n)$-vertex balanced bipartite (r,t)-RS graph where $r = (1/2 - \delta) \cdot n$ and $t = n^{\Omega(1/\log\log n)}$.*

Remark. We give a thorough description of the RS graph construction used to prove Proposition 2.3 in Section 5.3.2.

The Streaming Model

A data stream $\sigma = (e_1, e_2, \ldots, e_m)$ is a sequence of items where each item is from a large universe $\mathcal{U}$. The universe $\mathcal{U}$ depends on the underlying application or domain. In this thesis, we consider graph streams where the universe $\mathcal{U} = V \times V$ is the set of all possible edges of an n-vertex graph with the vertices V.

The goal of an algorithm is to compute a function of the stream σ while using space sublinear in the length of σ and size of $\mathcal{U}$. For example, this could be computing an approximate maximum matching in the underlying input graph G defined by σ while using only $O(n \operatorname{polylog} n)$ space, which is the focus of this thesis. An algorithm is allowed one or a few passes of the stream (in the same fixed order) to compute the function where the guarantees of the algorithm must hold for any ordering of the stream, especially an adversarial one[1].

A data stream σ may consist only of insertions (the insertion-only model) or both insertions and deletions (the insertion-deletion model or, equivalently, the dynamic streaming model). Depending on the underlying application or domain, there may be further restrictions on the structure of the stream.

[1] Other types of stream orders exist such as the random-order setting where the guarantees of the algorithm only need to hold for a uniform random ordering of the stream.

In this thesis, we consider graph streams that define an underlying input graph G, where edges do not have a negative value. Hence, graph streams may only delete an edge that has previously been inserted (and hasn't already been deleted). Furthermore, we consider graph streams that define a simple graph G and thus an edge may not be inserted multiple times (without first being deleted) and edges with the same two endpoints (self-loops) are never inserted.

Other variants of the streaming model exist – such as the *random-order* and *sliding winidow* models – and we refer the interested reader to the surveys in [Mut05] and [McG14] for further details.

The Communication Model

We consider the k-party randomised communication model for any integer $k \geq 2$, which is a natural generalisation of the two-party randomised communication model as defined by Yao [Yao79].

Let $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k, \mathcal{Z}$ be sets and $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_k \to \mathcal{Z}$ be a function. For each $i \in [k]$, player P_i is given the input $X_i \in \mathcal{X}_i$. The goal of the players is to compute $f(X_1, \ldots, X_k)$ by exchanging messages according to a randomised protocol π using public (shared) and private randomness. Let $\pi_{\text{out}}(X_1, \ldots, X_k)$ be the output of the protocol. π is called a δ-error protocol for function f if, for all possible inputs $(X_1, \ldots, X_k) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_k$,

$$\Pr[\pi_{\text{out}}(X_1, \ldots, X_k) = f(X_1, \ldots, X_k)] \geq 1 - \delta \ .$$

Definition 2.4. The *communication cost* of π, denoted $\mathsf{CCost}(\pi)$, is the maximum transcript length over all possible inputs and executions of the protocol.

Definition 2.5 (cf. [Yao79])**.** The δ-error *communication complexity* of f is the minimum communication cost over all δ-error protocols π for f, i.e., $\min_\pi\{\mathsf{CCost}(\pi)\}$.

We also work with a relatively more recent notion of *(internal) information cost* [CSWY01, BJKS02] of a protocol π. It represents the amount of *information* a player reveals about their input to another player via the transcript of messages communicated, which is taken over some distribution of the random variables. Furthermore, in this thesis, we only consider it w.r.t. two players.

Definition 2.6. Player P_1's information cost of protocol π is

$$\mathsf{ICost}^1_{\mathcal{D}}(\pi) = \mathcal{I}_{\mathcal{D}}(X_1; \Pi \mid X_2, R)$$

and, similarly, player P_2's information cost of π is

$$\mathsf{ICost}^2_{\mathcal{D}}(\pi) = \mathcal{I}_{\mathcal{D}}(X_2; \Pi \mid X_1, R)$$

where $\mathcal{I}_{\mathcal{D}}(A; B \mid C)$ denotes the mutual information of A and B conditioned on C when distributed according to distribution $\mathcal{D}$.

Remark. The sum of the players' (internal) information cost of a protocol is naturally a lower bound of its communication cost since every bit communicated reveals, at most, a single bit of information.

Connection to Graph Streaming. In this thesis, we consider graph problems in the communication model where the edges of a underlying input graph $G = (A, B, E)$ are arbitrarily partitioned among the players, that is, each X_i corresponds to edges of G. Furthermore, we consider protocols that communicate in a bounded number of rounds r and only allow communication among the k players in a one-way fashion. In particular, in each round of communication, the players communicate in order (starting with P_1) and can only send a message to the subsequent player (P_1 comes after P_k) except in the final round where the final player P_k returns the output instead.

These types of protocols are particularly interesting since any δ-error r-pass streaming algorithm that uses s space for any graph problem can be used to construct a δ-error r-round protocol for the same graph problem with communication cost $O(r \cdot k \cdot s)$ as follows:

- For each $i \in [k]$, player P_i fixes an arbitrary ordering σ_i of their input X_i.
- In each round $j \in [r]$ and for each player $i \in [k]$:
  - If $i, j = 1$, that is, player P_1 in the first round, then P_1 initialises the memory state of Alg. Otherwise, player P_i receives the memory state of Alg as a message from the previous player.
  - Player P_i then uses the memory state to continue running Alg on σ_i.
  - If $i \neq k$ and $j \neq r$, that is player P_i is not the final player in the final round, player P_i uses the updated memory state of Alg as their message in protocol π to the subsequent player. Otherwise, player P_k in the final round terminates Alg and uses its output as the output of protocol π.

Protocol π constructed in this way perfectly simulates r passes of Alg on the stream $\sigma = \sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_k$, which represents the underlying input graph G. Hence, protocol π succeeds whenever Alg succeeds. Furthermore, we have that each message communicated in π is the memory state of Alg that requires at most s bits. Since there are at most k messages communicated in each round and there are r rounds, at most $r \cdot k$ messages are communicated in total. This implies that the communication cost of π is $O(r \cdot k \cdot s)$.

The Query Model

In the query model of computation, an algorithm is only allowed access to the input via queries to an oracle. This is a general model of computation where the possible types of queries vary depending on the underlying application or domain.

When considering graph algorithms, as we do in this thesis, the types of queries broadly fall into two categories: (1) *local queries* such as vertex-degree queries, neighbourhood queries, and edge-existence queries [Fei06, GR08, Beh21], and (2) *global queries* such as (bipartite/maximal) independent set queries [BHR+20, AMM22, KOT24], linear/OR queries [ACK21], and maximal matching queries [KK20, KNS23]. Furthermore, queries may be made *adaptively* or *non-adaptively*. Adaptive queries are made in succession, and each query may be based on the responses of previous queries; hence, queries are made in rounds. Non-adaptive queries, on the other hand, are made in parallel and thus without knowledge of the responses from any of the queries.

In this thesis, we consider the query model where the goal of an algorithm is to learn a large matching in the underlying bipartite input graph $G = (A, B, E)$ via adaptive maximal matching edge-queries: In each query round i, an algorithm is allowed to make a single (adaptive) edge-query $Q_i \subseteq A \times B$ to an oracle that returns any maximal matching M_i in the edge-induced subgraph $G[E \cap Q_i]$, that is, the subgraph of G that contains the edges $E \cap Q_i$. Each query Q_i may be a function of $Q_1, \ldots, Q_{i-1}, M_1, \ldots, M_{i-1}$.

Query to Streaming Algorithms. Let $G = (A, B, E)$ be a bipartite graph presented as a stream of edges σ. Any r-round algorithm in the query model immediately implies an r-pass Greedy-only streaming algorithm for stream σ as follows: In each pass $i \in [r]$ of the stream, only the edges $e \in \sigma$ that are also in Q_i are fed to the Greedy algorithm to obtain a maximal matching M_i in the subgraph $G[E \cap Q_i]$, which perfectly simulates the oracle in each round i. However, the space used by the algorithm is dependent on the space required to represent the queries and, thus, on the specific algorithm considered. It is worth noting that this relationship holds in any model of computation where maximal matchings are generally easy to compute (e.g. the Massively Parallel Computation model [BHH19, GU19]), which is a key feature of query algorithms.

Query to Streaming Lower Bounds. Any lower bound on the approximation factor achievable in r rounds implies the same lower bound for r-pass Greedy-only streaming algorithms by making the oracle *streaming consistent*, which was initially highlighted in [KK20]. The goal is to construct a stream of edges σ such that, in each round $i \in [r]$, the oracle can be replaced by an execution of Greedy on the substream of σ corresponding to Q_i to exactly obtain the oracle's response M_i.

This property follows immediately by considering only queries that, in each round $i \in [r]$, *do not* include any edges of previously obtained matchings $M_1, \ldots, M_{i-1}$. Although this puts a restriction on the type of queries considered, this is not an issue. In particular, a query Q_i that *does* indeed include edges in $M_1, \ldots, M_{i-1}$ only gives the oracle the opportunity to reveal less information about the underlying graph $G = (A, B, E)$ by including a previously learned edge in its response M_i. Hence, Q_i can always be replaced by a query $Q'_i = Q_i \backslash (M_1 \cup \cdots \cup M_{i-1})$ that satisfies our restriction and is algorithmically better (or at least not worse). Finally, we can now construct the stream $\sigma = M_1 \circ M_2 \circ \cdots \circ M_r \circ F$ where F is the remainder of the edges of the underlying input graph $G = (A, B, E)$. It is then easy to see that the first edges processed by Greedy in the substream of σ corresponding to Q_i, in any round $i \in [r]$, is always going to be M_i. Since M_i is maximal in $G[Q_i \cap E]$, the output of Greedy is M_i as required.

3 Technical Overview

In this chapter, we present a detailed overview of the technical ideas required to prove our results for Greedy-only, Greedy-first, and arbitrary algorithms for approximate MBM in the insertion-only model and for arbitrary algorithms for approximate MVC in the insertion-deletion model.

3.1 Greedy-only Approximate MBM

In Chapter 4 (i.e., [KNS23]), we study the class of Greedy-only algorithms by considering the query model where an algorithm is allowed to make adaptive edge-queries to an oracle that returns a maximal matching in the corresponding edge-induced subgraph of the input graph (see Chapter 2 for details). By studying this model, we obtain an algorithm in three rounds/passes and lower bounds in one-, two-, and three rounds/passes. This completes our understanding of Greedy-only algorithms in one, two, and three passes: Doing better than a (1/2)-approximation requires at least three passes, where our (5/8)-approximation algorithm is optimal (regardless of the space used by the algorithm).

3.1.1 Three-Round/Pass Algorithm

Theorem 1. *There is an adaptive deterministic edge-query algorithm for MBM that achieves an* (5/8)*-approximation in three query rounds.*

The goal of any query algorithm for approximate MBM is to learn a large matching in the bipartite input graph $G = (A, B, E)$ with a maximum matching of size $\mu(G)$ where the algorithm initially only knows the set of vertices A and B. In each query round, the algorithm is allowed to make a single edge-query $Q \subseteq A \times B$ to an oracle that returns any maximal matching M in the edge-induced subgraph $G[E \cap Q]$, that is, the subgraph of G that contains the edges $E \cap Q$. Furthermore, each query Q is adaptive and thus may be based on the queries and responses from previous rounds.

Our algorithm builds on the three-round algorithm implied by the very first three-pass semi-streaming algorithm, which achieves a $(3/5 = 0.6)$-approximation [KT17] (see also [FKM+04,

KMM12]). This algorithm computes a maximal matching M_1 in the first round, then uses the second and third rounds to find a set of vertex-disjoint length-3 augmenting paths for M_1 – a strategy used by almost all two- and three-pass semi-streaming algorithms for approximate MBM (e.g. [EHM16, KT17, Kon18b, KN21, FS22, BKSW23]). Our algorithm improves this strategy.

Similar to previous work, we compute a maximal matching M_1 by querying all possible edges, that is, $Q_1 = A \times B$, in the first round and then use subsequent rounds to find augmenting paths for M_1 to increase its size. Since every augmenting path begins with a left wing in $H_L = G[A(M_1) \times B \backslash B(M_1)]$ and ends with a right wing in $H_R = G[A \backslash A(M_1) \times B(M_1)]$, in the second round, we obtain a maximal matching M_2 in $H_L \cup H_R$ by querying the corresponding edges as a single query, that is,

$$Q_2 = (A(M_1) \times B \backslash B(M_1)) \cup (A \backslash A(M_1) \times B(M_1)).$$

The edges of M_2 find left and right wings for the edges in M_1, and thus $M_1 \cup M_2$ consists of length-2 alternating paths (edges of M_1 with a single wing) and length-3 augmenting paths (edges of M_1 with both left and right wings). Let $A' \subseteq A(M_1)$ and $B' \subseteq B(M_1)$ be the endpoints of length-2 paths in $M_1 \cup M_2$. Current techniques in the literature would use the third round to find the missing wings that extend the length-2 alternating paths into length-3 augmenting paths. This is accomplished by finding edges from A' and B' to vertices unmatched by M_1, that is, by computing a maximal matching in $H'_L \cup H'_R$ where $H'_L = G[A' \times B \backslash B(M_1)]$ and $H'_R = G[A \backslash A(M_1) \times B']$. In fact, doing exactly this would achieve a (3/5)-approximation, which matches the approximation factor of the three-round algorithm implied by the very first three-pass semi-streaming algorithm.

Our key improvement in the third round is to also look for length-5 augmenting paths by finding edges that match A' to B'. In particular, any such edge combines two length-2 alternating paths into a length-5 augmenting path. Therefore, in the third round, we compute a maximal matching in the subgraph $H'_L \cup H'_{\text{in}} \cup H'_R$ where $H'_{\text{in}} = G[A' \times B']$ by querying the corresponding edges as a single query, that is,

$$Q_3 = (A' \times B \backslash B(M_1)) \cup (A' \times B') \cup (A \backslash A(M_1) \times B').$$

Each edge of M_3 thus completes either a length-3 or length-5 augmenting path. Although our algorithm finds many augmenting paths, they are not vertex-disjoint since a vertex unmatched by M_1 may be incident to wings found in both M_2 and M_3 (e.g. in Figure 3.1). That said, by considering a maximum independent set in the intersection multi-graph of the augmenting paths, we are able to argue that at least half of them form a vertex-disjoint set. Overall, we show that our algorithm finds a large enough set of vertex-disjoint augmenting paths to always output an (5/8)-approximation for MBM. See Figure 3.1 for an example run of our algorithm.

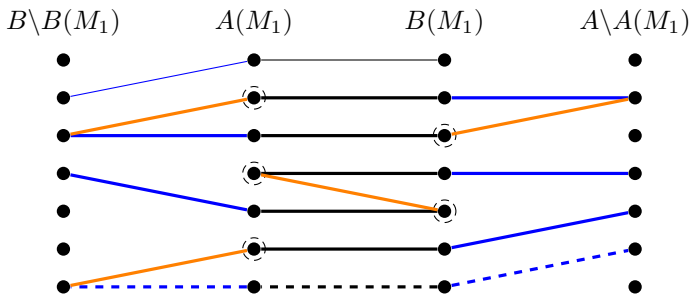


Figure 3.1: An illustration of a run of our three-round algorithm. The edges of M_1 are black, the edges of M_2 are blue, and the edges of M_3 are orange. The vertices of A' and B' used in the third round query are circled. The dashed edges represent an augmenting path found in the second round, while the thick edges represent augmenting paths found in the third round.

Proving our result naturally relies on bounding the size of the maximal matchings returned in each round of the algorithm, which then implies the number of augmenting paths found as a function of the size of the initial maximal matching M_1. We accomplish this by fixing a maximum matching M^* and counting the number of its edges that appear in each queried subgraph. A key challenge, however, is to track the interaction of the edges of M^* and the edges of the maximal matchings returned in each round.

To that end, a crucial property we require from our choice of M^* is that the symmetric difference $M_1 \oplus M^* = (M_1 \backslash M^*) \cup (M^* \backslash M_1)$ has no even-length alternating paths or cycles, that is, it only contains augmenting paths. This allows us to partition the vertices of $A(M_1)$ and $B(M_1)$, respectively, into vertices that are endpoints of M^* edges that either (1) start/end an augmenting path, (2) are in the middle of an augmenting path, or (3) are not part of any augmenting path, that is, are already optimal $M_1 \cap M^*$ edges. Each part of the partition affects the sizes of the returned maximal matchings in different ways and thus carefully tracking these effects allows us to obtain a tight bound on the number of augmenting paths found.

3.1.2 One-, Two-, and Three-Round/Pass Lower Bounds (Greedy-Only)

Theorem 2. *There does **not** exist an adaptive deterministic edge-query algorithm for MBM that achieves a better than*

1. $(1/2)$*-approximation in one query round,*
2. $(1/2 + o(1))$*-approximation in two query rounds, or*
3. $(5/8 + o(1))$*-approximation in three query rounds.*

To prove lower bounds in the query model, we consider a game between a player (the algorithm) and the maximal matching oracle. The goal of the player is to learn a large matching

in an underlying bipartite graph $G = (A, B, E)$ that is adversarially constructed by the oracle along the way.

The player initially only knows the vertices A and B and is allowed to query the oracle with any set of edges $Q \subseteq A \times B$ in each round, typically basing the query on any information about G revealed in previous rounds. The oracle then returns an adversarially chosen maximal matching M in the subgraph $G[E \cap Q]$, revealing as little information about a large matching as possible. In each round, the player's query Q reveals both *edges* and *non-edges* about the graph G. In particular, the edges M are definitely in G, and the edges of Q that are not incident to M are definitely not in G due to the maximality of M. Throughout the game, once information about G is revealed, it is fixed and thus may not be altered in subsequent rounds.

The underlying technique behind our lower bounds is to keep track of the information revealed by the oracle whenever it returns a maximal matching in response to a query, which was first employed by Khalil and Konrad [KK20] for vertex-queries. The main challenge for a multi-round lower bound is to show that the oracle is able to respond to *any* sequence of queries made by the player while keeping hidden a large enough matching so that the player's output is *not* a good enough approximation. Therefore, the complexity of the problem naturally increases as the number of rounds increases. Although these ideas are employed by Khalil and Konrad [KK20] for vertex-queries, our arguments are substantially different as we consider the more general edge-query model, which adds a further layer of complexity. To tackle this, similar to [KK20], we make simplifying assumptions to reduce the complexity of the queries (without affecting the hardness of the problem). In addition to that, our arguments require carefully partitioning the queries in order to obtain structural properties than can be exploited.

Simplifying assumptions. In each round i, a query Q_i made by the player forces the oracle to reveal edges M_i and non-edges N_i about the underlying graph that is being constructed. There are, however, some queries that are better than others in the sense that they learn more edges and non-edges. To simplify our arguments, we show that, regardless of the query Q_i, no more than a certain set of edges $\tilde{M}_i$ and non-edges $\tilde{N}_i$ (up to isomorphisms) are revealed to the player, which we denote as the hard structure graph $\tilde{H}_i = (A, B, \tilde{M}_i, \tilde{N}_i)$ – determining the appropriate edges and non-edges are crucial to obtaining our lower bounds in each round. This allows us to assume that, at the end of round i, the player knows $\tilde{H}_i$, that is, the edges $\tilde{M}_i$ and non-edges $\tilde{N}_i$. Therefore, when considering the subsequent round $i+1$, the complexity of the queries in rounds $1, 2, \ldots, i$ are entirely considered by $\tilde{H}_i$.

Query partitioning. The simplifying assumptions alone are, however, not sufficient. In particular, in rounds $i \geq 2$, a single structure graph $\tilde{H}_i$ of edges and non-edges is not enough to handle the complexity of the edge-queries. Therefore, using the structural properties of any query Q_i received, we are able to partition the vertices such that the corresponding portion

of Q_i in each vertex-induced partition has desirable properties that can be exploited. In particular, it allows us to obtain a single hard structure graph for each partition. Although this potentially creates multiple different hard structure graphs to deal with separately in the subsequent round $i+1$, it provides a simplification of the arguments. That said, we are able to simplify this further in our second round arguments by partitioning the query in such a way that each partition has the same hard structure graph $\tilde{H}_2$. Hence, in the third round, we only have to consider $\tilde{H}_2$ to entirely consider the complexity of the queries in the first two rounds.

Our lower bounds. In each of our lower bound arguments, the underlying input graph that the oracle ultimately constructs is a $(2n)$-vertex balanced bipartite graph G that contains a maximum matching of size $\mu(G) \approx n$, and in the first round, G indeed has a perfect matching of size $\mu(G) = n$.

First round. Let Q_1 be any arbitrary query in the first round. The oracle computes a maximum matching $M^*(Q_1)$ among the query edges Q_1. If $|M^*(Q_1)| \leq n/2$, namely, it is no better than a $(1/2)$-approximation, then the maximal matching $M_1 = M^*(Q_1)$ is returned to the player who ultimately learns no non-edges. If $|M^*(Q_1)| > n/2$, then the oracle picks an arbitrary subset $M_1 \subseteq M^*(Q_1)$ of size exactly $n/2$ and returns it to the player who certainly learns some non-edges that have endpoints unmatched by M_1. Therefore, the edges M_1 and non-edges N_1 learned by the player, regardless of the query Q_1, are always going to be at most the edges of a matching $\tilde{M}_1$ of size $n/2$ and the non-edges $\tilde{N}_1 = A_{\text{out}} \times B_{\text{out}}$ that make $\tilde{M}_1$ maximal where $A_{\text{out}} = A \backslash A(\tilde{M}_1)$ and $B_{\text{out}} = B \backslash B(\tilde{M}_1)$. Although this is sufficient for any Q_1, in order to simplify our arguments in the subsequent rounds, we further assert that $\tilde{M}_1$ is an induced matching by adding the non-edges $(A_{\text{in}} \times B_{\text{in}}) \backslash \tilde{M}_1$ to $\tilde{N}_1$ where $A_{\text{in}} = A(\tilde{M}_1)$ and $B_{\text{in}} = B(\tilde{M}_1)$. We now have our hard structure graph $\tilde{H}_1$ where $\tilde{M}_1$ is no better than a $(1/2)$-approximation – see Figure 3.2 for an illustration.

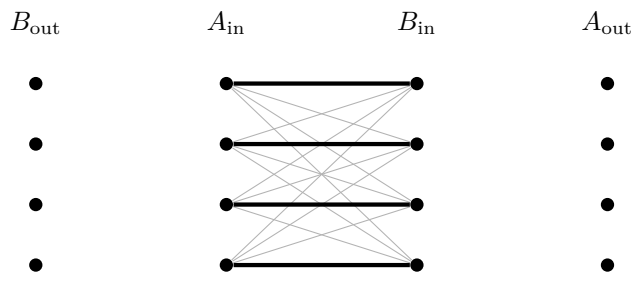


Figure 3.2: An illustration of structure graph $\tilde{H}_1$. The solid black edges represent the matching $\tilde{M}_1$. The non-edges between A_{out} and B_{out} are implicit by the layout of the vertices, and the remaining non-edges in $\tilde{N}_1$ are the grey edges. Any edges not drawn are potential edges in the underlying input graph that the player hasn't learned.

Second round. Let Q_2 be any arbitrary query in the second round. Using $\tilde{H}_1$ as our starting point, we can assume that the player's query does not include the edges in $\tilde{M}_1$ and $\tilde{N}_1$ since doing so can only reduce the edges and non-edges learned in this round. With that, the oracle computes a maximum matching $M^*(Q_2)$ among the query edges Q_2. Since the edges of $M^*(Q_2)$ can only be in $A_{\text{in}} \times B_{\text{out}}$ or $A_{\text{out}} \times B_{\text{in}}$, they either form length-2 alternating paths or length-3 augmenting paths with $\tilde{M}_1$. We thus partition the vertices in such a way that the length-2 and length-3 paths can be considered separately. By similar arguments used in the first round, in either partition, the player only learns at most the edges of a matching $\tilde{M}_2$ of size $\approx n/2$ that essentially only forms length-2 alternating paths with $\tilde{M}_1$ and at most the non-edges $\tilde{N}_2$ that make it maximal. This means that no query Q_2 can find augmenting paths with $\tilde{M}_1$ and thus can not do better than a $(1/2 + o(1))$-approximation[1]. Again, although this is sufficient for any Q_2, in order to simplify the third round arguments, we assert that $\tilde{M}_2$ is induced by adding the relevant non-edges to $\tilde{N}_2$. We now have the same hard structure graph $\tilde{H}_2$ in each partition. See Figure 3.3 for an illustration.

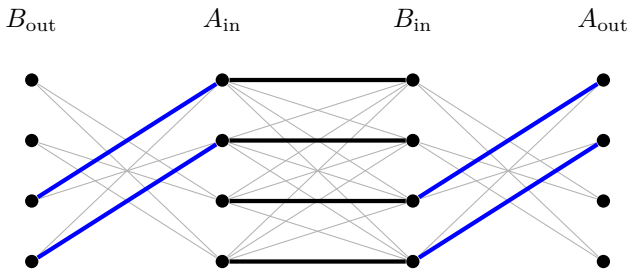
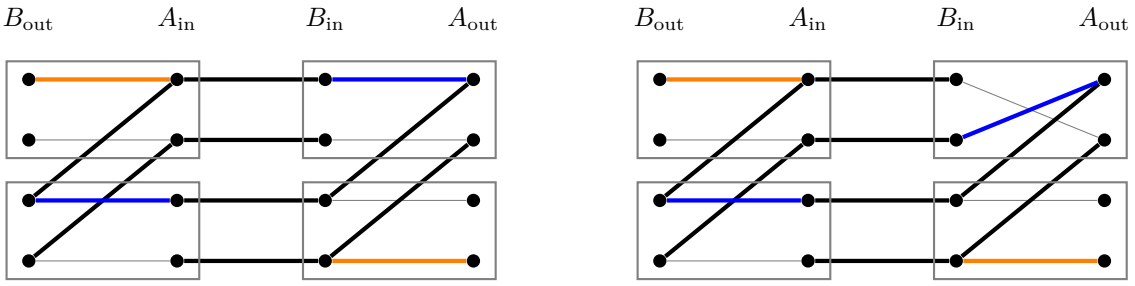


Figure 3.3: An illustration of the structure graph $\tilde{H}_2$. The black edges are $\tilde{M}_1$ and the blue edges are $\tilde{M}_2$. The grey edges are the non-edges $\tilde{N}_2$ (not including the ones implicit due to the structure). Any edges not drawn are potential edges in the underlying input graph that the player hasn't learned.

Third round. Let Q_3 be any arbitrary query in the third round and let P denote the length-2 paths in $\tilde{M}_1 \cup \tilde{M}_2$. From our three-round algorithm, we know that there is some query Q_3 that finds augmenting paths where half the paths in P extend into length-3 augmenting paths, namely, to obtain an $(5/8)$-approximation. Observe that this can be achieved if every path is extended into length-3 augmenting paths, but in a way where each augmenting path can uniquely be paired with another one to form a 6-cycle. To that end, we partition the vertices into two types of 16-vertex gadgets where, in each gadget, either the query edges Q_3 necessarily form 6-cycles as described above, or they necessarily do not. The gadgets with query edges that form 6-cycles are simple to deal with since we can easily construct a maximal matching $\tilde{M}_3$ with corresponding non-edges $\tilde{N}_3$ where $\tilde{M}_1 \cup \tilde{M}_2 \cup \tilde{M}_3$ in each gadget forms a 6-cycle. The other

[1]The $o(1)$ term comes from considering a case where $|\tilde{M}_1|$ is odd.

gadgets are also simple to deal with since not having the necessary query edges to form 6-cycles implies a rigid structure among the query edges Q_3 in the gadget. Therefore, we exploit this to obtain a maximal matching $\tilde{M}_3$ and corresponding non-edges $\tilde{N}_3$ that reveals only a single length-3 augmenting path. In each type of gadget, we obtain a different hard structure graph $\tilde{H}_3^{\circ}$ and $\tilde{H}_3^{\overline{\circ}}$, respectively. However, in both hard structure graphs, the maximum matching in $\tilde{M}_1 \cup \tilde{M}_2 \cup \tilde{M}_3$ is no better than an $(5/8 + o(1))$-approximation. See Figure 3.4 for an illustration.



(a) The hard structure graph $\tilde{H}_3^{\circ}$.

(b) The hard structure graph $\tilde{H}_3^{\overline{\circ}}$. The non-edges, in this case, may vary slightly.

Figure 3.4: An illustration of the hard structure graphs obtained in the third round on each 16-vertex gadget. The black edges are $\tilde{M}_1$ and the blue edges are $\tilde{M}_2$. The grey boxes are drawn to simplify the presentation of the non-edges where all the edges not contained in a box are non-edge. Non-edges are further represented by grey edges. Any edges not drawn are potential edges in the underlying input graph that the player has not learned.

3.2 Greedy-first Approximate MBM

In Chapter 5 (i.e., [KN21]), we study the class of Greedy-first semi-streaming algorithms for approximate MBM, which includes all semi-streaming algorithms that solely compute a Greedy maximal matching of the input graph in the first pass of the stream. We give a new two-pass algorithm that matches the current best algorithm and obtain the first lower bound for two-pass Greedy-first algorithms, which significantly narrows the approximation factor gap for two-pass Greedy-first semi-streaming algorithms to $[2 - \sqrt{2}, 2/3]$.

3.2.1 Two-Pass Algorithm

Theorem 3. *For $p \in (0, 1]$ and integer $d \geq 1$, there is a two-pass streaming algorithm for MBM with approximation factor*

$$\begin{cases} \frac{1}{2} + (\frac{1}{d+p} - \frac{1}{2d}) \cdot p, & \text{if } p \leq d(\sqrt{2}-1) \\ \frac{1}{2} + \frac{d-p}{6d+2p}, & \text{otherwise,} \end{cases}$$

(ignoring lower order terms) that succeeds with probability at least $1 - 1/\operatorname{poly}(n)$ and uses $O(n \log n)$ bits of space.

Given a bipartite input graph $G = (A, B, E)$ with a maximum matching M^* of size $\mu(G)$ that is presented as a stream of edges σ, the goal of any two-pass semi-streaming algorithm for approximate MBM is to output a large matching in two passes of the stream using $O(n \operatorname{polylog} n)$ space where it initially only knows the vertices A and B.

Our algorithm is a two-pass Greedy-first algorithm and thus solely uses the first pass to compute a maximal matching M. In the second pass, we aim to find a large set of vertex-disjoint augmenting paths for M, particularly length-3 augmenting paths. This is a natural strategy for two-pass semi-streaming algorithms [EHM16, KT17, Kon18b, BKSW23] and has been successful in achieving a better than (1/2)-approximation.

Trivial two-pass algorithm. Consider first the trivial two-pass Greedy-first algorithm that, given a maximal matching M computed in the first pass, uses the second pass to obtain Greedy maximal matchings M_L and M_R in the subgraphs $H_L = G[A(M) \cup B \backslash B(M)]$ and $H_R = G[A \backslash A(M) \cup B(M)]$, respectively. A hard instance for this algorithm is when G has a perfect matching M^* (of size $\mu(G) = n$) and M is of size $n/2$, namely, M is exactly a (1/2)-approximation. Then, the symmetric difference $M \oplus M^* = (M \backslash M^*) \cup (M^* \backslash M)$ is exactly a set of $n/2$ vertex disjoint length-3 augmenting paths and thus H_L and H_R also have perfect matchings. Therefore, in a worst-case ordering of the edges of the stream, M_L obtains left wings for exactly half the set of edges in M, and M_R obtains right wings for the other half of the edges in M. Therefore, only length-2 alternating paths are found, and M remains a (1/2)-approximation. See Figure 3.5 for an illustration.

Our meta-strategy. In order to improve this algorithm, we want M_L and M_R to somehow find left and right wings, respectively, for the same edges in M, which would thus find length-3 augmenting paths. Naturally, a one-pass algorithm that achieves a better than (1/2)-approximation would be sufficient, but accomplishing this has been an elusive open question for two decades. Fortunately, to obtain the very first better than (1/2)-approximation in two passes, Konrad, Magniez, and Mathieu [KMM12] introduced a deterministic and a randomised

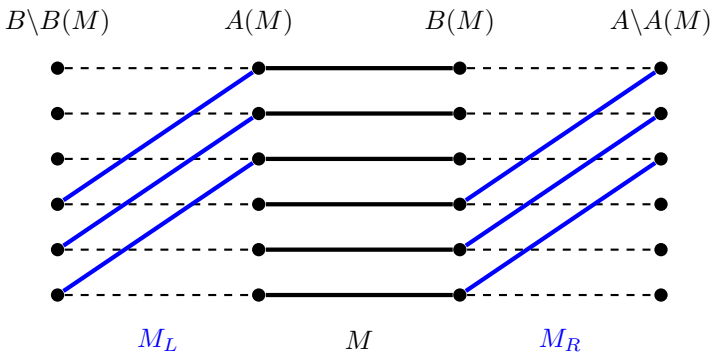


Figure 3.5: An illustration of the trivial two-pass algorithm. The black edges are M_1. The blue edges on the left correspond to left wings M_L, and the blue edges on the right correspond to right wings M_R. The dashed black edges correspond to M^*.

strategy that both solve this problem by sacrificing the size of the matching obtained. Their strategies are as follows:

- Run Greedy$_d$ on σ to obtain a *semi-matching* S, that is, greedily compute a set of edges S such that, for any $a \in A$ and $b \in B$, $\deg_S(a) \leq 1$ and $\deg_S(b) \leq d$ (or vice versa) for any integer $d \geq 1$.
- Sample each vertex in A with probability $p \in (0, 1]$ to obtain A' and let $H = G[A' \cup B]$ be the subgraph of edges incident to a vertex in A'. Run Greedy on σ_H to obtain a matching M_H where σ_H is the substream of σ containing only the edges of H.

In our work, we combine both of their strategies and obtain the following meta-strategy:

- Sample each vertex in A with probability $p \in (0, 1]$ to obtain A' and let $H = G[A' \cup B]$. Run Greedy$_d$ on σ_H to obtain a semi-matching S_H such that, for any $a \in A$ and $b \in B$, $\deg_S(a) \leq 1$ and $\deg_S(b) \leq d$ for any integer $d \geq 1$.

Note that by either fixing $p = 1$ or $d = 1$ in our meta-strategy, we get back the deterministic or randomised strategies in [KMM12], respectively. By fixing both $p = 1$ and $d = 1$, we get back the Greedy algorithm.

To highlight the strength of our meta-strategy (and the strategies in [KMM12]), let $G = (A, B, E)$ be a bipartite input graph with a maximum matching M^* of size $\mu(G)$ that is presented as a stream of edges σ. Consider first the Greedy algorithm. Whenever Greedy adds an edge $e \in \sigma$ to its initially empty maximal matching M, each of its endpoints may block an edge of the (fixed) maximum matching M^* from being added to M, totalling at most two blocked edges in M^*. Then, by a simple counting argument, the size of M is at least half the size of M^* – this is precisely why Greedy is a $(1/2)$-approximation. Consider now our

meta-strategy in the same light, that is, an instance where GREEDY$_d$ adds an edge $e = ab \in \sigma_H$ to its initially empty semi-matching S_H.

We still have that the endpoint $a \in A$ may block an edge $ab^* \in M^*$ that is necessarily also in H since $e = ab \in \sigma_H$ implies that $a \in A'$, that is, $ab^* \in M^* \cap H$. This is, however, no longer the case for the endpoint $b \in B$. Although the endpoint $b \in B$ is now matched in S_H, it may be matched at most d times in S_H, and thus it only contributes a $1/d$ fraction to blocking the incident edge $a^*b \in M^*$ from being added to S_H. Furthermore, the edge a^*b is only in H with probability p since the random choice of whether $a^* \in A'$ has not yet been considered, which means that vertex b may not even partially block an edge in $M^* \cap H$. Therefore, the endpoint b may block a p/d fraction of an edge in $M^* \cap H$ in expectation. Overall, the edge e that is added to S_H blocks at most $1 + p/d$ edges of $M^* \cap H$ in expectation and we have that

$$\mathbb{E}[|S_H|] \geq (d/(p+d)) \cdot \mathbb{E}[|M^* \cap H|] = (d/(p+d)) \cdot p \cdot \mu(G).$$

To obtain this result, similar to [KMM12] for their randomised strategy, we use a charging argument where each edge of S_H potentially charges (or blocks) edges in $M^* \cap H$ – our measure of progress is substantially different than [KMM12]. However, since $|S_H|$ is a random variable and the potential charge is also a random variable, we end up taking the expectation over $|S_H|$ many random variables where $|S_H|$ is a random variable itself. Hence, to evaluate this, we apply Wald's Equation. We then strengthen the result using concentration bounds, namely, Azuma-Hoeffding's inequality, to obtain the result with high probability, that is, with probability at least $1 - 1/\operatorname{poly}(n)$.

Our meta-algorithm. We obtain our two-pass semi-streaming meta-algorithm by improving the aforementioned trivial two-pass algorithm using our meta-strategy in a similar way to previous work that either uses the deterministic [EHM16, KT17] or randomised [Kon18b] strategies in [KMM12].

We replace the computation of M_L in H_L and M_R in H_R in the trivial algorithm by computing the semi-matchings S_L in H_L and S_R in H_R using our meta-strategy. By ensuring that the vertices in $A(H_L) = A(M)$ and $B(H_R) = B(M)$ have at most one incident edge in the semi-matchings S_L and S_R, we only find at most one left and one right wing for each edge in M. This implies that our strategy also needs to consider a random subset of the vertices in $A(M)$ and $B(M)$, respectively. If the sampling for each set of vertices is done independently, the subset of edges in M that S_L finds left wings for may not align with the subset of edges of M that S_R finds right wings for. Hence, similarly to [Kon18b], we coordinate the random sampling by sampling the edges of M with probability p to obtain a random subset M' and then use $A(M') \subseteq A(M)$ and $B(M') \subseteq B(M)$ as the random subsets for the computation of S_L and S_R, respectively.

For simplicity of exposition, suppose again that the input graph G has a perfect matching and the initial maximal matching M is exactly a $(1/2)$-approximation, which implies that H_L and H_R also have perfect matchings (see Figure 3.5). After running our algorithm using any set of parameters where $p < 1$ or $d \geq 2$, we are guaranteed that more than half of M' have left *or* right wings. By the pigeonhole principle, there are edges in M' that are guaranteed to have both left *and* right wings, thus length-3 augmenting paths are found. The augmenting paths, however, are no longer vertex-disjoint since each vertex in $A \backslash A(M)$ or $B \backslash B(M)$ may have up to d incident augmenting paths. By considering the intersection multi-graph of the augmenting paths and finding a maximum independent set, we can obtain a set of vertex-disjoint augmenting paths that contains at least a $1/d$ fraction of all the augmenting paths found. This can then be used to increase the size of the initial maximal matching M. See Figure 3.6 for an example run of our meta-algorithm on a hard graph instance – full details of the hard instance are given in Section 5.4.2.

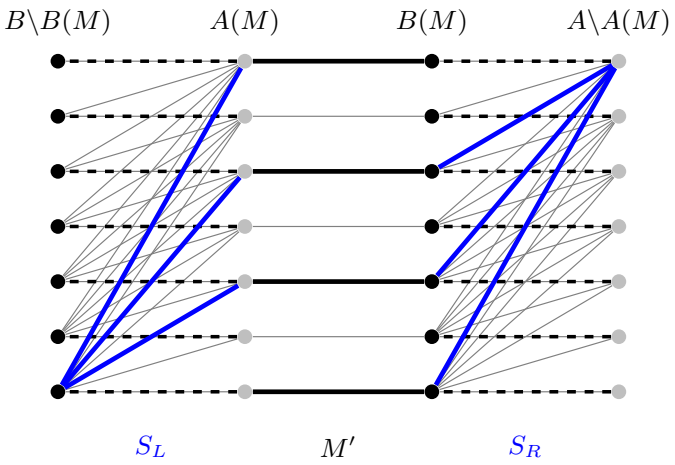


Figure 3.6: An illustration of our two-pass algorithm on a hard instance. The black edges are M'_1. The blue edges on the left are the semi-matching S_L, and the blue edges on the right are the semi-matching S_R. The dashed black edges correspond to M^*. The grey edges are the remaining edges of the hard instance.

The number of vertex-disjoint augmenting paths found by our algorithm, and thus its approximation factor, is optimised for two parameter settings: (1) For $d = 1$ and $p = \sqrt{2} - 1$, we obtain the current best two-pass semi-streaming algorithm by Konrad [Kon18b] that uses only the randomised strategy in [KMM12]; and (2) For $d = 2$ and $p = 2\sqrt{2} - 2$, we obtain a new algorithm that combines both the deterministic and randomised strategies in [KMM12]. We also obtain a hard graph instance and ordering of its edges that proves our analysis is tight for any choice of $d \geq 1$ and $p \leq d \cdot (\sqrt{2} - 1)$. Finally, our meta-algorithm only stores M, S_L, and S_R – each of which has $O(n)$ many edges – and so it uses the optimal $O(n \log n)$ space.

3.2.2 Two-Pass Lower Bound (Greedy-First)

Theorem 4. *For constant $\epsilon > 0$, any constant-error (possibly randomised) Greedy-first streaming algorithm for MBM that achieves a $(2/3+\epsilon)$-approximation in two passes requires $n^{1+\Omega(1/\log\log n)}$ bits of space.*

Streaming lower bounds are typically obtained by proving lower bounds in the communication model due to the well-known fact that any streaming algorithm for a problem can be used to obtain a communication protocol for the same problem where the space used by the algorithm directly implies the communication cost of the protocol (see Chapter 2 for details).

In our work, we consider the problem of computing an approximate MBM in the two-player one-round communication model where the players Alice and Bob are each given a disjoint subset of edges E_A and E_B that make up the underlying bipartite graph $G = (A, B, E_A \cup E_B)$. Then, allowing only a single message (one round) to be communicated from Alice to Bob, Bob must output a large matching. Typically, one-round communication lower bounds in this setting imply space lower bounds for one-pass arbitrary algorithms for approximate MBM. However, by exploiting the very restricted first pass of the stream for Greedy-first algorithms, we alter this setting to obtain a setting where one-round communication lower bounds imply space lower bounds for two-pass Greedy-first algorithms for approximate MBM.

To that end, by placing a maximal matching M at the start of the stream, we can guarantee that the Greedy maximal matching computed in the first pass learns only the edges of M, that is, the second pass of the algorithm begins with knowledge of the maximal matching M. Hence, by augmenting Alice and Bob's inputs with the edges of the maximal matching M, they both have all the edges M that possibly could have been learned in the first pass of a Greedy-first algorithm and thus the second pass of such an algorithm corresponds to a protocol in this augmented communication model. We now have that a lower bound on the communication cost of all protocols in this setting implies a lower bound on the space of the second pass of a Greedy-first algorithm, which is sufficient to prove a lower bound on the space for two-pass Greedy-first algorithms.

We obtain our hard graph construction in the augmented two-party communication model for Greedy-first algorithms by building on the hard graph construction used in [GKK12], which we discuss first.

[GKK12]'s hard graph construction. Goel, Kapralov, and Khanna [GKK12] show that (r,t)-*Ruzsa-Szemerédi* (RS) graphs [RS78] – graphs whose edge set is made up of t many edge-disjoint induced matchings of size r – are crucial for constructing hard graph constructions for MBM. In particular, RS graphs are extremely useful due to the following property:

(RS) For any pair of induced matchings M_1, M_2 of an RS graph, the edges of M_1 do not reveal

anything about the subgraph induced by the vertices of M_2 since M_2 is an induced matching and is edge-disjoint from M_1.

The hard graph construction in [GKK12] uses an (r, t)-RS graph with linear sized matchings, that is, $r = \Theta(n)$, by uniform randomly selecting an induced matching M_s to be special in a way that any 'large' matching of the graph requires learning a linear sized matching in the subgraph induced by the vertices of M_s. Since M_s is a linear-sized induced matching, the subgraph induced by its vertices is defined to only have the edges of M_s, and therefore, a constant fraction of its edges are part of any 'large' matching. In their construction, the random choice of M_s is identified by a separate set of $\Theta(n)$ many selector edges, which is a matching that matches all vertices not in $V(M_s)$ to an entirely new set of vertices. In particular, Alice is given the edges of the RS graph, and Bob is given the selector edges. See Figure 3.7 for an illustration.

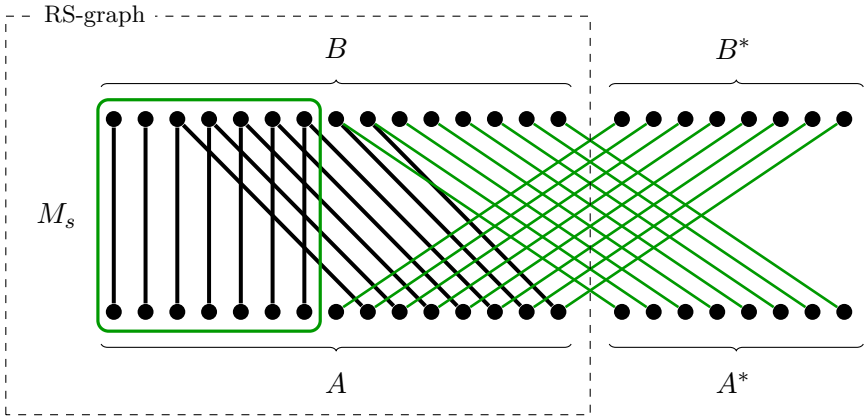


Figure 3.7: An illustration of [GKK12]'s hard graph construction on vertices $A \cup A^*$ and $B \cup B^*$. The black dashed box represents the RS graph (with black edges) used in their construction on vertices A and B. The green edges to the new set of vertices A^* and B^* represent the selector edges. The green box highlights the special induced matching M_s.

The hardness of their construction comes from the fact that, in order to output a 'large' matching, Alice needs to communicate a constant fraction of the special induced matching. However, without knowledge of which one is special, Alice essentially needs to communicate a constant fraction of *all* induced matchings due to Property (RS). Then, by obtaining an (r, t)-RS graph construction with $t = n^{\Omega(1/\log\log n)}$ and $r \approx n/2$ (see also [FLN+02]) and using this in their hard graph construction, they show that a 'large' matching is one that is better than a $(2/3 \approx 0.667)$-approximation and the communication required to achieve it is $\Omega(r \cdot t) = n^{1+\Omega(1/\log\log n)}$ bits, which is strictly more than semi-streaming space.

Remark 3.1. The communication lower bound comes from the fact that Alice needs to com-

municate a constant fraction of each induced matching. Interestingly, communicating even a small constant fraction of each induced matching is essentially as hard as communicating the entire induced matching. Therefore, outputting a 'large' matching is just as hard as outputting a maximum matching in their construction.

Our hard graph construction. We build on the hard graph construction in [GKK12] by obtaining a maximal matching M that is no better than a $(2/3)$-approximation and whose knowledge does not affect the hardness of the graph construction – since both Alice and Bob will know these edges. We observe first that a perfect matching in the RS graph used in their graph construction satisfies the properties that it is no better than a $(2/3)$-approximation and that it is maximal in the entire graph. However, constructing such a matching that doesn't affect the hardness is more nuanced.

To that end, we study the (r,t)-RS graph that is used in their construction and give a similar construction with $t = n^{\Omega(1/\log\log n)}$ many edge-disjoint induced matchings of size $r \approx n/2$ and, additionally, with $t/2$ many edge-disjoint (non-induced) matchings of size $2r \approx n$, that is, they are near-perfect matchings (Theorem 5). For each induced matching M_i, for $i \in [t]$, in their RS graph construction, we are able to add a symmetric copy M_i' that matches an entirely different set of vertices. We show that this matching is also an induced matching, and overall, we effectively double the number of induced matchings in the RS graph. The key property we obtain is that, although $M_i \cup M_i'$ is not an induced matching, it is still a matching that is twice the size of M_i, that is, $|M_i \cup M_i'| \approx n$.

We obtain our hard graph construction by using our (r,t)-RS graph with near-perfect matchings in the hard graph construction in [GKK12]. By arbitrarily picking an induced matching M_i, for $i \in [t]$, and its symmetric copy M_i', along with a few additional edges to make it a perfect matching in the RS graph, we obtain the desired maximal matching M that is given to both Alice and Bob. Note that the special induced matching M_s is picked uniformly at random from the remaining induced matchings so that knowledge of M does not affect the hardness of the graph construction due to Property (RS). The hardness shown in [GKK12] then follows in our setting by a reduction, which proves that any better than a $(2/3)$-approximation requires $n^{1+\Omega(1/\log\log n)}$ bits of communication in our setting for Greedy-first algorithms. This then implies our lower bound for two-pass Greedy-first semi-streaming algorithms.

Comparison to lower bounds for Greedy-only algorithms. Despite the lower bounds for Greedy-first and Greedy-only algorithms both being for restricted classes of algorithms, the techniques used to prove them are entirely disjoint.

The techniques that we introduce for proving Greedy-first lower bounds follow more closely to the standard techniques for proving arbitrary lower bounds, that is, we use an augmented two-party communication model instead of the standard two-party communication model. This

is due to the fact that only the first pass of GREEDY-first algorithms are restricted whereas the subsequent passes are unrestricted (like arbitrary algorithms). On the other hand, we prove GREEDY-only lower bounds in the query model, which requires techniques that are entirely disjoint from standard ones for arbitrary algorithms. This is due to the fact that every pass of the algorithm has the same restriction that can be modeled using a single adaptive maximal matching query.

3.3 Arbitrary Approximate MBM

In Chapter 6 (i.e., [KN24]), we study arbitrary algorithms that use semi-streaming space. We obtain the first unconditional lower bound for two-pass arbitrary semi-streaming algorithms, which we accomplish by using entirely different techniques to previous multi-pass streaming lower bounds for exact and approximate MBM [GO13, AR20, CKP+21, KN21, Ass22, AS23a] and other graph problems (see [Ass23] for a recent overview). In doing so, we narrow the approximation factor gap for two-pass arbitrary semi-streaming algorithms to $[2-\sqrt{2}, 8/9]$.

3.3.1 Two-Pass Lower Bound (Arbitrary)

Theorem 6. *For constant $\epsilon > 0$, any constant-error (possibly randomised) two-pass streaming algorithm for MBM that achieves a $(8/9+\epsilon)$-approximation requires $n^{1+\Omega(1/(\log\log n)^2)}$ bits of space.*

As previously mentioned, streaming lower bounds are typically obtained by proving lower bounds in the communication model due to the well-known fact that any streaming algorithm for a problem can be used to obtain a communication protocol for the same problem where the space used by the algorithm directly implies the communication cost of the protocol (see Chapter 2 for details). Although there has been significant progress in proving one-round communication lower bounds for approximate MBM (implying one-pass streaming lower bounds) [GKK12, Kap13, Kap21], these results do not extend to two rounds as proving communication lower bounds for two-round protocols introduces additional obstacles.

To highlight this, consider again the graph construction in [GKK12] that is hard in one round. Alice holds the $n^{1+\Omega(1/\log\log n)}$ many edges of an RS graph, and Bob holds $\Theta(n)$ many selector edges that make one of Alice's induced matchings special in a way that a constant fraction of its edges is needed to output a large matching. Although this is hard for one-round protocols, it becomes easy for two-round protocols since the selector edges are 'easy to learn'. In particular, in the first round, it is sufficient for Bob to communicate his $\Theta(n)$ selector edges to Alice, and then in the second round, Alice knows the entire underlying input graph and thus trivially knows a maximum matching, which can easily be sent to Bob using $\Theta(n)$ edges.

Overall, this requires $O(n \log n)$ bits of communication to output a maximum matching, not just an approximation.

Our hard graph construction. We build on [GKK12]'s graph construction and overcome the hurdle with the 'easy to learn' selector edges by 'hiding' a special collection of *many* smaller instances of [GKK12]'s graph construction among *many* similar collections. The novelty of our construction comes from using RS graphs to structure the collections such that each is an *induced* collection of many *vertex-disjoint* instances.

To that end, we use a *global* $(2n_g)$-vertex balanced bipartite (r_g, t_g)-RS graph G_g^{RS} with linear sized matchings as a template to embed *local* instances of [GKK12]'s bipartite graph construction. More precisely, we first replace each of the $2n_g$ vertices of G_g^{RS} with a vertex group of size $\Theta(n_\ell)$ and then replace each of the $r_g \cdot t_g$ edges of G_g^{RS} with a bipartite local instance on the corresponding pair of vertex groups. The resulting graph is thus a bipartite graph with $r_g \cdot t_g$ many edge-disjoint local instances where each local instance is a balanced bipartite graph on $\Theta(n_\ell)$ vertices. In particular, each local instance is made up of an (r_ℓ, t_ℓ)-RS graph with linear sized matchings and $\Theta(n_\ell)$ many selector edges (as in [GKK12]). By using an RS graph as a template, we crucially have that each of the t_g many induced matchings of G_g^{RS} now represents an induced collection of r_g vertex-disjoint local instances, that is, the subgraph induced by the vertex groups of the collection is exactly the vertex-disjoint union of its local instances (it contains no other edges that may 'corrupt' an instance). Furthermore, by Property (RS), we have that learning the edges of one induced collection does not reveal anything about the edges of another induced collection.

In order to identify a special collection in our construction, similar to [GKK12], we define a separate set of $\Theta(n)$ many *global* selector edges that match all the vertices (of the vertex groups) that *do not* belong to the special induced collection to a new set of vertices. Therefore, we define a 'large' matching as one that requires finding more than just the global selector edges and local selector edges in the special induced subgraphs. Using the N-vertex (r, t)-RS graph with $t = N^{\Omega(1/\log\log n)}$ matchings of size $r \approx N/2$ in [GKK12] as our global and local RS graphs, a 'large' matching corresponds to a better than $(8/9 \approx 0.889)$-approximation. See Figure 3.8 for an illustration. Finally, the edges of our graph construction are then partitioned between three players (instead of two as in [GKK12]). In particular, Alice receives the edges of the $r_g \cdot t_g$ many local (r_ℓ, t_ℓ)-RS graphs and Bob receives the $r_g \cdot t_g$ many sets of $\Theta(n_\ell)$ local selector edges where each one selects an induced matching M_s to be special in one of Alice's RS graphs (recall [GKK12]'s construction). The third player, Charlie, receives the $\Theta(n)$ global selector edges.

The hardness of our construction comes from the fact that learning the global selector edges in the first round, followed by learning the local selector edges of the special collection in the second round only obtains an $(\approx 8/9)$-approximate MBM. However, doing better requires

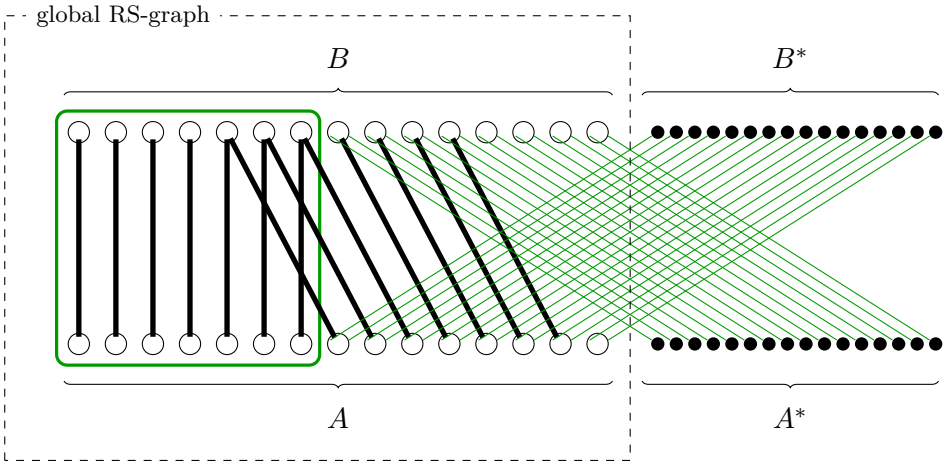


Figure 3.8: An illustration of our hard two-pass graph construction. The unfilled circular nodes represent the vertex groups that partition the vertices A and B, each of which represents a vertex in G_g^{RS}. The thick (black) edges between the vertex groups, which exist for every edge $(u, v) \in G_g^{\text{RS}}$, represent pairs of vertex groups (A_u, B_v) that have local edges between them – these edges are illustrated in Figure 3.7. A thick (green) box highlights the special global induced matching, which represents the special induced collection. The remaining (green) edges to the new set of vertices A^* and B^* represent the global selector edges.

learning a linear number of edges of the r_g many special induced matchings M_s, each of size r_ℓ, of the special collection, which is essentially the same as obtaining a better than (2/3)-approximation in a constant fraction of the special local instances. We formally prove that doing better than a (8/9)-approximation is hard by abstracting the hardness of our construction as a new communication problem HiddenStrings.

Remark 3.2. Similar to Remark 3.1 for [GKK12]'s construction, learning a constant fraction of the edges of the special induced matchings is just as hard as learning all of them, and thus, obtaining a better than (8/9)-approximation in our hard graph construction is just as hard as obtaining a maximum matching, that is, learning all r_ℓ edges of each of the r_g many special induced matchings of the special collection.

A new communication game. HiddenStrings is defined as a three-party two-round communication problem that is parameterised by the integers $t_g, r_g, t_\ell, r_\ell \geq 1$. The players, Alice, Bob, and Charlie, are given the following inputs that correspond to our hard graph construction G:

- For each $i \in [t_g], j \in [r_g], k \in [t_\ell]$, Alice is given a local bit string $X[i, j, k] \in \{0, 1\}^{r_\ell}$ that corresponds to one of the $t_g \cdot r_g \cdot t_\ell$ many induced matchings of size r_ℓ held by Alice in G;
- For each $i \in [t_g], j \in [r_g]$, Bob is given a local index $K[i, j] \in [t_\ell]$ that corresponds to the identity of a special matching M_s in a local instance (or RS graph) and thus the local selector edges held by Bob in G; and

- Charlie is given a global index $I^* \in [t_g]$ that corresponds to the identity of the special collection of local instances and thus the global selector edges held by Charlie in G.

The goal is to output the r_ℓ-bit strings $X[I^*, j, K[I^*, j]]$ for all $j \in [r_g]$, which corresponds to the edges of the r_g many special induced matchings M_s, each of size r_ℓ, in the special collection – this captures the hardness of our hard graph construction G due to Remark 3.2. See Figure 3.9 for an illustration of the HiddenStrings problem.

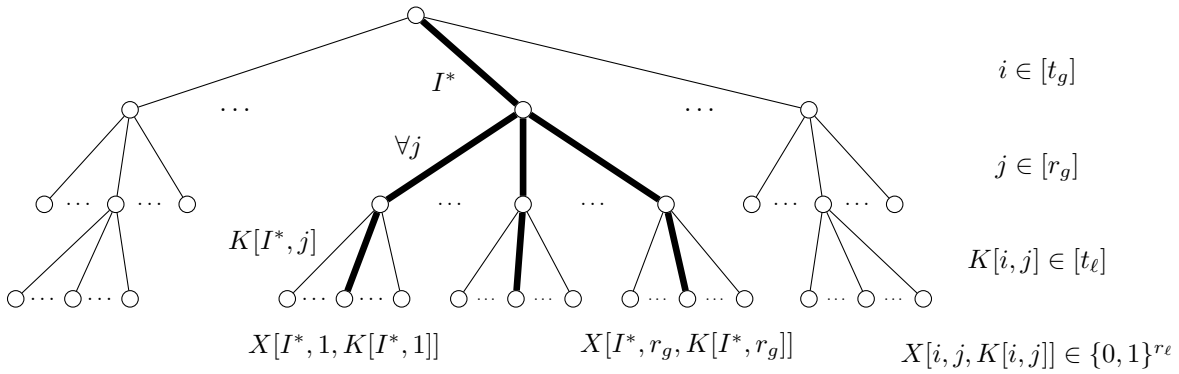


Figure 3.9: An illustration of the HiddenStrings problem. Alice's input strings $X[i, j, k]$ are arranged in a tree structure where the index i corresponds to the second level, j to the third level, and k to the fourth level (the first level is the root). The strings $X[i, j, k]$ are located at the leaves of the tree. The objective is to output the strings located at the leaves in the subtree consisting of the bold edges, i.e., the strings $X[I^*, j, K[I^*, j]]$, for all $j \in [r_g]$.

HiddenStrings can be seen as a generalisation of the Index problem. In fact, at the heart of our proof for the two-round communication lower bound for HiddenStrings is the information cost tradeoff result for Index by Jain, Radhakrishnan, and Sen [JRS09], which we first discuss.

Information cost tradeoff for Index. In the Index_N problem, Alice is given a bit string $Y \in \{0,1\}^N$, Bob is given a special index $J \in [N]$, and the goal is to output the special bit $Y[J]$. Consider the input distribution $\mathcal{D}_{\mathsf{I}}$ where Y and J are distributed independently and uniformly. If communication by a protocol π_{I}^1 for Index_N is restricted to a single message from Alice to Bob (one round), then it is easy to show that Alice's message must reveal $\mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(Y; \Pi_{\mathsf{I}} \mid J, R_{\mathsf{I}}) = \Omega(N)$ bits of information about her input to Bob (e.g. [FNSZ23, Lemma C.4]). This is called Alice's (internal) information cost and is denoted $\mathsf{ICost}^A_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}^1) = \mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(Y; \Pi_{\mathsf{I}} \mid J, R_{\mathsf{I}})$ where Π_{I} represents the transcript of messages communicated by the protocol π_{I}^1, R_{I} represents the public random bits used by protocol π_{I}^1, and $\mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(A; B \mid C)$ denotes mutual information of A and B conditioned on C distributed according to $\mathcal{D}_{\mathsf{I}}$. Bob's (internal) information cost is defined similarly as $\mathsf{ICost}^B_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}^1) = \mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(J; \Pi_{\mathsf{I}} \mid Y, R_{\mathsf{I}})$.

On the other hand, when a protocol is allowed even just two rounds, the following protocol

π_{I}^2 highlights an exponential tradeoff: In the first round, Bob sends the b most significant bits of his index J to Alice, who then communicates only the $N/2^b$ entries of Y that are consistent with Bob's message in the second round. Note that, since a single bit communicated carries at most a single bit of information, the information cost of a protocol is a lower bound on its communication cost and thus $\mathsf{ICost}^B_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}^2) \leq b$ and $\mathsf{ICost}^A_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}^2) \leq N/2^b$. Jain, Radhakrishnan, and Sen [JRS09] prove that this tradeoff is tight, even for protocols that are allowed any number of rounds of communication.

Proposition 3.3 (cf. [JRS09])**.** *Let $b \leq O(\log N)$ be an integer, and let $\delta < \frac{1}{2}$ be a positive constant. Any δ-error protocol π_{I} for Index_N is such that:*

1. *Either $\mathsf{ICost}^A_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) = \mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(Y; \pi_{\mathsf{I}} \mid J, R_{\mathsf{I}}) \geq \frac{N}{2^{O(b)}}$, or*
2. *$\mathsf{ICost}^B_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) = \mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(J; \pi_{\mathsf{I}} \mid Y, R_{\mathsf{I}}) \geq b$,*

where $\mathcal{D}_{\mathsf{I}}$ denotes the uniform distribution and R_{I} is the public randomness used by π_{I}.

HiddenStrings Lower Bound. We consider two-round protocols for HiddenStrings where communication occurs as follows:

$$\underbrace{\text{Alice} \longrightarrow \text{Bob} \longrightarrow \text{Charlie} \longrightarrow}_{\text{Round 1}} \underbrace{\text{Alice} \longrightarrow \text{Bob} \longrightarrow \text{Charlie} \longrightarrow}_{\text{Round 2}} \text{output}\ .$$

In particular, there are two natural protocols for HiddenStrings:

1. In the first round, both Bob and Charlie forward their inputs in entirety and then, in the second round, Alice can trivially solve the problem and forward the solution to Charlie (through Bob), who then outputs it. Overall, the communication cost of this protocol is $\tilde{O}(t_g \cdot r_g + r_g \cdot r_\ell)$ bits, where the additive $O(r_g \cdot r_\ell)$ is just to communicate the solution.
2. In the first round, only Charlie forwards the global index I^* and then, in the second round, Alice forwards the bit strings $X[I^*, j, k]$ for all $j \in [r_g], k \in [t_\ell]$. Bob can now solve the problem and forward the solution to Charlie, who then outputs it. Overall, the communication cost of this protocol is $\tilde{O}(r_g \cdot t_\ell \cdot r_\ell)$ bits.

Depending on the choice of the parameters used in HiddenStrings, one of these protocols will be better in the sense that its communication cost is smaller. In particular, the first protocol is better when $t_g \ll t_\ell \cdot r_\ell$. We prove that these natural protocols are best possible (up to logarithmic factors) by showing that the communication cost of any two-round protocol is $\Omega(\min\{t_g \cdot r_g, r_g \cdot t_\ell \cdot r_\ell\})$ bits (Theorem 7).

To prove this result, we show that any protocol π_{HS} for HiddenStrings with parameters t_g, r_g, t_ℓ, r_ℓ can be used to construct a protocol π_{I} for Index_{t_ℓ} where Alice's input $Y \in \{0,1\}^{t_\ell}$ and Bob's input $J \in [t_\ell]$ are independently and uniformly distributed according to $\mathcal{D}_{\mathsf{I}}$. Then,

using the information cost tradeoff result for Index (Proposition 3.3), we show that π_{HS} either reveals $\Omega(t_g \cdot r_g)$ bits about K (as in the first natural protocol) or $\Omega(r_g \cdot r_\ell \cdot t_\ell)$ bits about X (as in the second natural protocol).

Protocol π_{I}. Alice and Bob each use their own private randomness in π_{I} to sample their HiddenStrings inputs uniformly and independently according to $\mathcal{D}_{\mathsf{HS}}$, namely, the inputs X and K, respectively. Then, using their shared public randomness in π_{I}, they sample $I^* \in [t_g]$ (Charlie's input), $J^* \in [r_g]$, and $\varphi^* \in [r_\ell]$. This allows both players to simulate the missing player Charlie and also appropriately embed Y and J as follows: Alice sets $X[I^*, J^*, k][\varphi^*] = Y[k]$, for each $k \in [t_\ell]$, and Bob sets $K[I^*, J^*] = J$. This ensures that the output of π_{HS} contains $Y[J]$ and the players' inputs are still distributed according to $\mathcal{D}_{\mathsf{HS}}$. Finally, Alice and Bob run π_{HS} in two rounds where, in the second round, Alice simulates first-round Charlie in π_{HS} and Bob simulates second-round Charlie in π_{HS} – this is a perfect simulation of π_{HS}. Bob learns the output of π_{HS}, which is $X[I^*, j, K[I^*, j]]$, for every $j \in [r_g]$, and then returns $X[I^*, J^*, K[I^*, J^*]][\varphi^*](= Y[J])$ as the output for π_{I}. Therefore, π_{I} succeeds whenever π_{HS} succeeds.

Analysis. For clarity, we will refer to Alice and Bob in the simulation of π_{HS} as $\text{Alice}_{\mathsf{HS}}$ and Bob_{HS}. The key to our analysis is to assume that π_{HS} communicates at most $O(t_g \cdot r_g)$ bits in the first round, that is, Bob_{HS} reveals at most $O(t_g \cdot r_g)$ bits about K. Then, since the uniform random choices of I^* and J^* are unknown to Bob_{HS} before communicating his first message, the special index $J(= K[I^*, J^*])$ held by Bob_{HS} is uniformly hidden among his $t_g \cdot r_g$ indices in K. Then, by a direct-sum-like argument, we show that Bob_{HS}'s first message, which is thus Bob's message in π_{I}, reveals $O(1)$ bits of information about the special index $J = K[I^*, J^*]$. Since Bob only communicates once, this means that $\mathsf{ICost}^B_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) = O(1)$. The information cost tradeoff result for Index (Proposition 3.3) then implies that Alice in π_{I} must reveal $\Omega(t_\ell)$ bits about Y, namely, $\mathsf{ICost}^A_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) = \Omega(t_\ell)$. However, since the uniform random choices of J^* and φ^* remain unknown to $\text{Alice}_{\mathsf{HS}}$ throughout the entire simulation of π_{HS}, the embedding of $Y[k]$ as $X[I^*, J^*, k][\varphi^*]$ for every $k \in [t_\ell]$ allows us to show, using a direct-sum-like argument, that $\text{Alice}_{\mathsf{HS}}$ essentially sends all the strings $X[I^*, j, k]$, for every $j \in [r_g]$ and $k \in [t_\ell]$, in the second round, that is, $\text{Alice}_{\mathsf{HS}}$ in π_{HS} reveals $\Omega(r_g \cdot t_\ell \cdot r_\ell)$ bits about X.

MBM to HiddenStrings. We now complete the two-round lower bound for approximate MBM. Using the already established one-to-one correspondence between HiddenStrings and our hard graph construction, we show that any protocol π_{MBM} that achieves an $(8/9 + \varepsilon)$-approximate MBM, for any constant $\varepsilon > 0$, with communication cost s can be used to solve HiddenStrings, which proves our main result by the well-known connection to the streaming model (see Chapter 2 for details).

For simplicity of exposition, we consider inputs (X, K, I^*) of HiddenStrings that are uniformly and independently distributed according to $\mathcal{D}_{\mathsf{HS}}$. Recall first that each edge held by Alice corresponds to a uniform random bit in X. If we naively construct the underlying graph such that the edges held by Alice are also uniformly present (if its bit is 1) or absent (if the bit is 0), then we would obtain a graph that is expected to have half the edges of each local induced matching. Although this would still give us a lower bound result, the quality of a 'large' matching would be worse. We overcome this challenge by only deleting Alice's edges with probability 2ε if it corresponds to a 0 bit. In particular, each edge is deleted with probability ε in expectation since the bits are uniform.

Consider now the output of π_{MBM}. Since it finds an $(8/9+\varepsilon)$-approximation, it is guaranteed to find a $\Theta(\varepsilon)$ fraction of the r_g many special induced matchings M_s, each of size r_ℓ, in the special collection. This corresponds to learning edges that correspond to a linear fraction of the bits in $X[I^*, j, K[I^*, j]]$, for all $j \in [r_g]$, which is the desired output for HiddenStrings. Naturally, this is not sufficient since (1) all the bits need to be output (not just a linear fraction), and (2) edges present may correspond to either a 1 bit or a 0 bit (with different probabilities) that needs to be distinguished.

We tackle both of these issues by running $O(\text{polylog}\, n/\varepsilon)$ instances of π_{MBM} in parallel on inputs $(X', K', I^{*\prime})$ that, to each run of π_{MBM}, look as though they have been sampled independently from $\mathcal{D}_{\mathsf{HS}}$, but still represent the same underlying input (X, K, I^*). This can be accomplished by appropriately XOR-ing the bits of X with uniform random bits that are generated using public randomness and appropriately permuting the positions of X with uniform random permutations that are, again, generated using public randomness – doing this with public randomness keeps this unknown to each run of π_{MBM} while allowing the operations to be undone given the output of each run of π_{MBM}.

Finally, considering $O(1/\varepsilon)$ runs of π_{MBM} together allows us to find all the edges (at least once) that correspond to each and every bit in $X[I^*, j, K[I^*, j]]$, for all $j \in [r_g]$. Then, considering $O(\text{polylog}\, n)$ of these groupings allows us to find each edge corresponding to each bit $O(\text{polylog}\, n)$ times, which allows us to distinguish the 1 and 0 cases. Therefore, we successfully solve HiddenStrings using π_{MBM} with only a $O(\text{polylog}\, /\varepsilon)$ overhead in space. For the appropriate choices of parameters t_g, r_g, t_ℓ, r_ℓ, we obtain a strictly better than semi-streaming communication lower bound for HiddenStrings, which then translates to a similar lower bound for approximate MBM since, for constant ε, the $O(\text{polylog}\, /\varepsilon)$ overhead does not significantly affect the quality of the lower bound[2].

[2]Note that ε must be constant due to the density of RS graphs with linear-sized matchings used in the construction [GKK12].

3.3.2 Related Work

Comparison to previous work. Assadi [Ass22] give a (conditional) lower bound for arbitrary two-pass semi-streaming algorithms for approximate MBM. Starting with the hard graph construction in [GKK12], they overcome the hurdle with the 'easy to learn' selector edges by replacing them with alternating paths in a clever way using XOR gadgets that are themselves based on RS graphs – a technique that has previously been used in works for proving lower bounds for multi-pass algorithms for exact MBM [AR20, CKP+21] (see also [AN21, AB21]). In their construction, learning the selector edges now requires essentially storing all the edges of the XOR gadgets in the first pass, which requires more than semi-streaming space. Therefore, the identity of the special induced matching is still unknown at the beginning of the second pass and outputting a 'large' matching essentially requires storing the entire RS graph as in [GKK12], which also requires more than semi-streaming space.

Their construction successfully rules out certain approximation factors for MBM in two passes using semi-streaming space; however, the size of a 'large' matching is conditioned on the maximum density of RS graphs with linear-sized matchings. While only $n/2^{\Omega(\log^* n)}$ many matchings have been ruled out [FHS17], the densest known constructions have $n^{\Omega(1/\log\log n)}$ many matchings [FLN+02, GKK12]. In the best case scenario, their results rule out better than (≈ 0.98)-approximations[3], whereas they do not rule out any constant factor approximations using known RS graph constructions.

Very recently, Assadi and Sundaresan [AS23a] extended these ideas to multiple passes. By building on the ideas in [CKP+21], they show how to construct the XOR gadgets in [Ass22] using permutations that are hard to find even when many passes are allowed. With that, they rule out better than $(1-\varepsilon)$-approximations in $o(\log(1/\varepsilon))$ passes using semi-streaming space. However, similar to [Ass22], the size of a 'large' matching, in this case, the range of ε, is conditioned on the maximum density of RS graphs with linear-sized matchings. Under the assumption that RS graphs with $n^{\Omega(1)}$ linear sized matchings exist, ε is bounded by a small constant, whereas it is only bounded by $o(1)$ using known RS graph constructions.

Relations to follow up work. The issue with the 'easy to learn' selector edges faced in [GKK12]'s graph construction is a more general issue with hard one-pass graph constructions (e.g. [CDK19, ACK19b, Kap21, DK20]). The maximal independent set problem (MIS) is one such problem where it is known that $\Omega(n^2)$ bits of space is required in one pass [CDK19, ACK19b], and until very recently [AKNS24], lower bounds in just two passes were not known.

Assadi et al. [AKNS24] show that our idea of hiding *many* independent instances of a hard one-pass graph construction can be used to obtain a multi-pass hard graph construction

[3]Small constant factors are ruled out under the assumption that RS graphs with $n^{\Omega(1)}$ linear sized matchings exist.

for MIS. Using RS graphs, however, is no longer possible since the hard one-pass graph construction for MIS is not bipartite, i.e., 2-colourable. They take our idea a step further and resolve this issue by obtaining an appropriate generalisation of RS graphs. They construct a family of graphs that have many 'strongly' induced collections of paths of length k, which thus allows for $(k+1)$-colourable hard graph constructions[4]. With that, they obtain a hard graph construction for MIS in each pass for a sufficiently large number of passes, thus obtaining a hierarchical family of hard graph constructions for MIS.

By tackling this problem in the communication model, they obtain a multi-pass/round analysis of the hierarchical family of hard graph constructions for MIS using round-elimination and, crucially, message compression. This improves on our use of the information cost tradeoff for Index, which is limited to two rounds. Their results imply that $\Omega(\log\log n)$ passes are required by any semi-streaming algorithm for MIS, which matches the $O(\log\log n)$-pass semi-streaming algorithm (up to constant factors).

We observe that the hardness for their construction can also be abstracted as HiddenStrings. Hence, their analysis techniques would also be able to prove a multi-round communication lower bound for HiddenStrings. This would then imply a multi-pass space lower bound for approximate MBM. In particular, it would show that any p-pass algorithm that achieves a better than $(3^p/(3^p-1))$-approximation for MBM requires $\Omega(n^{1+\Omega(1/(\log\log)^p)})$ space, that is, it would rule out $(1-\varepsilon)$-approximations in $o(\log(1/\varepsilon))$ passes.

3.4 Insertion-Deletion Approximate MVC

In our (student-only) work in Chapter 7 [NS22], we study arbitrary algorithms for approximate MVC in insertion-deletion streams. We obtain a one-pass algorithm that obtains any α-approximation using the optimal space up to constant factors. This gives us a complete understanding of the problem in one-pass in the insertion-deletion model for all possible space regimes, not just semi-streaming space.

3.4.1 One-Pass Algorithm

Theorem 8. *There is a one-pass insertion-deletion streaming algorithm for α-approximate MVC that succeeds with probability at least $1-1/\operatorname{poly}(n)$ and uses $O(n^2/\alpha^2)$ bits of space for any $\alpha \leq n^{1-\delta}$ where $\delta > 0$.*

Let $G=(V,E)$ be any n-vertex graph with a minimum vertex cover V^* of size $\mathrm{vc}(G)$ that

[4]Strongly induced in this contexts means that, in addition to the edges of the paths being the only edges present in the collection's corresponding vertex-induced subgraph, there are also no shortcuts of the paths that use vertices outside of the subgraph. Note that, for RS graphs (with collections of paths of length one), the latter property trivially holds.

is presented as a stream σ consisting of both edge insertions and deletions. The goal of any one-pass streaming algorithm for α-approximate MVC (αMVC) is to output a vertex cover of size at most $\alpha \cdot \text{vc}(G)$ in one pass of the stream σ using as little space as possible.

It has been shown by Dark and Konrad [DK20] that $\Omega(n^2/\alpha^2)$ bits is necessary for αMVC. They also give a simple deterministic algorithm using $O((n^2/\alpha^2)\cdot\log\alpha)$ bits of space, matching the lower bound up to a logarithmic factor. Their algorithm arbitrarily partitions the vertices into n/α groups of size α and uses counters, each requiring $O(\log\alpha)$ bits, to maintain the number of edges between each of the $\Theta(n^2/\alpha^2)$ pairs of vertex groups. Then, at the end of the stream, they compute a group-level minimum vertex cover and then return the vertices of the covering groups, which forms an α-approximate solution.

Our algorithm. We improve the algorithm in [DK20] by first computing the partition of the vertices uniformly at random and maintaining the appropriate $\Theta(n^2/\alpha^2)$ many counters while processing the stream. Additionally, while processing the stream, we maintain as many random edge samplers as possible using $O(n^2/\alpha^2)$ bits of space – typically, each sampler requires $O(\log^3 n)$ bits of space to randomly sample an edge. Then, at the end of the stream, we run Greedy on the randomly sampled edges of the underlying input graph G. The key idea is that the sparseness properties implied by the random Greedy matching are able to reduce the space required by the counters. In particular, processing a random set of edges of a graph using Greedy to obtain a matching M is well-known in the streaming literature to imply sparseness properties in the residual subgraph induced on the vertices not matched by M, i.e., $G[V\backslash V(M)]$ [ACG+15, Kon18a, GKMS19, AS22]. However, the sampling strategy used affects the quality of our result.

Uniform sampling. Uniformly sampling s edges from the graph G and obtaining a Greedy matching M implies that the subgraph induced by residual subgraph has a maximum degree bound of $\tilde{O}(|E|/s)$ [ACG+15, Kon18a, GKMS19], which is independent of the size of the matching M obtained. This sparseness property is due to the fact that uniform sampling is skewed towards sampling high-degree vertices. Hence, M either matches these vertices or has the chance to match many of their neighbours, which decreases their residual degree. By sampling as many uniform edges as possible using $O(n^2/\alpha^2)$ bits of space[5], we obtain a $\text{poly}(\alpha)$ maximum degree bound in the residual subgraph, which is tight even when considering the average degree of the residual subgraph[6].

The next simple but crucial observation is that the vertices of $V(M)$ cover all the edges that are not in the residual subgraph. Furthermore, it only uses at most twice as many vertices

[5]Each edge can be sampled using an ℓ_0 sampler that uses $O(\log^3 n)$ bits of space [JST11, KNP+17]

[6]Consider a graph with a large clique on $\Theta(n/\alpha)$ vertices where almost all the edges are sampled from, and many smaller cliques which assert the guaranteed max degree bounds and thus essentially the same average degree bound.

that would have been required to cover these edges due to the well-known property that V^* must include at least one vertex of each edge in M. Now, we have that adding all vertices $V(M)$ to the output vertex cover will not affect the overall approximation factor. Hence, it remains only to consider the residual subgraph, which we know is sparse due to the poly(α) maximum degree bound. Then, since the vertex groups are randomly partitioned, maintaining the number of edges between any pair of vertex groups now only requires counters that use less space than in [DK20]'s algorithm. In fact, for $\alpha \ll n^{1/4}$, each of the $\Theta(n^2/\alpha^2)$ many counters uses $O(1)$ bits of space on average. Then, computing the group-level minimum vertex cover as before and combining it with $V(M)$ obtains an α-approximation using $O(n^2/\alpha^2)$ bits of space.

Non-uniform sampling. To obtain our algorithm for the full range of approximation factors α, we necessarily obtain our random set of edges for computing the Greedy matching M by non-uniformly sampling the edges using *neighbourhood edge sampling*, that is, first sampling a group of vertices and then uniformly sampling from the neighbourhood of the group – this biases away from high degree vertices and is able to obtain a better average degree bound. This strategy was successfully used for approximate MM in [AS22], which they call the Match-or-Sparsify algorithm. After appropriately altering their algorithm for our purposes, the key strength of Match-or-Sparsify is that it *either* gives us a large matching M of size $\Omega(n/\alpha)$ ('easy' case) *or* implies an average degree bound of $\tilde{O}(1/\alpha)$ in the residual subgraph ('hard' case)[7]. In the 'easy' case, M is large enough to imply that an optimal solution V^*, which must include at least one vertex of each edge in M, is of size $\Omega(n/\alpha)$ and thus the trivial vertex cover V is an ($\approx \alpha$)-approximate solution. In the 'hard' case, we are in the same situation as before, except we now have a stronger average degree bound, which is sufficient for our previous arguments to hold for the full range of approximation factors α.

As previously mentioned, although we use the Match-or-Sparsify algorithm in [AS22], its guarantees are not sufficient for our purposes without some alterations. To that end, we first discuss their algorithm and then present our alterations.

Match-or-Sparsify. For some parameter $\beta \leq n$, the Match-or-Sparsify$_\beta$ algorithm in [AS22] non-uniformly samples edges using space $O(\beta^2/\alpha^3)$ bits, and then computes a Greedy matching from them. They give an intricate analysis to show that their algorithm either finds a large matching of size at least $\beta/8\alpha$ or implies that the residual subgraph has at most $20 \cdot \beta \cdot \log^4 n$ edges [AS22, Lemma 16]. Unlike uniform sampling, the residual properties (sufficiently) only hold when the matching is small – a key property exploited in their analysis. Additionally, in order for the guarantees to hold, they rely on the assumption that $\beta \geq \alpha^2 \cdot n^\delta$. Informally, when β is set as the size of the maximum matching μ ($= \mu(G)$), Match-or-Sparsify$_\mu$ finds a large matching in 'easy' graph cases and a sparse residual subgraph in the 'hard'

[7] Recall that α here corresponds to the approximation factor of MVC and is thus such that $\alpha > 1$.

graph cases. However, μ is not known, so they find a setting of β close to μ by running $\mathsf{Match\text{-}or\text{-}Sparsify}_\beta$ in parallel with β as all powers of 2 between 1 and n.

Our Alterations. The first thing to note is that the 'easy' and 'hard' instances for MM and MVC are not the same. Consider $\mathsf{Match\text{-}or\text{-}Sparsify}_\beta$ when a large Greedy matching is found. Since at least one endpoint of each matching edge must be in a vertex cover, it implies that $\text{vc}(G) \geq \frac{\beta}{8\alpha}$. However, returning a solution to αMVC at this stage can only be of size at most $\Theta(\beta)$, which would not be a trivial solution (the entire vertex set) with $\beta \ll n$. Furthermore, we have no guaranteed sparseness properties since the matching found is large. Hence, instead of needing $\beta \approx \mu$, which requires $\log n$ many runs to find, we only need a single run of $\mathsf{Match\text{-}or\text{-}Sparsify}_n$ (with the parameter β fixed as n). Secondly, their assumption that $\beta \geq \alpha^2 \cdot n^\delta$ implies that $\alpha \leq n^{\frac{1-\delta}{2}}$, but we require it to hold for any $\alpha \leq n^{1-\delta}$. Since we have an additional α factor of space (see [DK20]), we can increase the number of non-uniformly sampled edges to use $O(n^2/\alpha^2)$ bits instead, which allows us to remove the assumption. Finally, the increase in the number of samples also allows us to improve the sparseness guarantees of the residual subgraph by an α factor.

4 Greedy-Only Approximate MBM

This chapter has been published as the following work:

- [KNS23]: Christian Konrad, Kheeran K. Naidu, and Arun Steward. Maximum Matching via Maximal Matching Queries. In *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023*.

Abstract

We study approximation algorithms for Maximum Matching that are given access to the input graph solely via an *edge-query maximal matching oracle*. More specifically, in each round, an algorithm queries a set of potential edges and the oracle returns a maximal matching in the subgraph spanned by the query edges that are also contained in the input graph. This model is more general than the *vertex-query model* introduced by binti Khalil and Konrad [FSTTCS'20], where each query consists of a subset of vertices and the oracle returns a maximal matching in the subgraph of the input graph induced by the queried vertices.

We give tight bounds for deterministic edge-query algorithms for up to three rounds. In more detail:

1. As our main result, we give a deterministic 3-round edge-query algorithm with approximation factor 0.625 on bipartite graphs. This result establishes a separation between the edge-query and the vertex-query models since every deterministic 3-round vertex-query algorithm has an approximation factor of at most 0.6 [binti Khalil, Konrad, FSTTCS'20], even on bipartite graphs. Our algorithm can also be implemented in the semi-streaming model of computation in a straightforward manner and improves upon the state-of-the-art 3-pass 0.6111-approximation algorithm by Feldman and Szarf [APPROX'22] for bipartite graphs.
2. We show that the aforementioned algorithm is optimal in that every deterministic 3-round edge-query algorithm has an approximation factor of at most 0.625, even on bipartite graphs.

3. Last, we also give optimal bounds for one and two query rounds, where the best approximation factors achievable are $1/2$ and $1/2+\Theta(\frac{1}{n})$, respectively, where n is the number of vertices in the input graph.

4.1 Introduction

In this chapter, we study approximation algorithms for the Maximum Matching problem (MM) and its bipartite version, the Maximum Bipartite Matching problem (MBM), that are only given query access to the input graph $G = (V, E)$ via a *maximal matching oracle* – we call this the *edge-query model.* In each round i of the maximal matching edge-query model, the algorithm sends a set of potential edges $Q_i \subseteq V \times V$, denoted the query edges, to the oracle, which in turn responds with an arbitrary maximal matching in the subgraph $G[Q_i \cap E]$, i.e., the subgraph of G spanned by the query edges that are also contained in G.

Maximal Matching Queries The study of algorithms for MM that solely execute maximal matching queries was introduced by binti Khalil and Konrad [KK20]. They considered a *vertex-query model*, where, in each round i, the algorithm queries a subset of vertices $U_i \subseteq V$, and the oracle responds with an arbitrary maximal matching in subgraph $G[U_i]$, i.e., the subgraph induced by vertices U_i. The edge-query model is more general than the vertex-query model since vertex queries can be simulated in the edge query model: For each query $U_i \subseteq V$ in the vertex-query model, querying the set of edges Q_i that turns U_i into a clique yields an equivalent edge-query algorithm.

The study of maximal matching query models is motivated by the fact that, in many computational models, including the data streaming model [McG14] and the Massively Parallel Computation model [KSV10], computing maximal matchings is easy, while computing substantially larger matchings is more challenging. Computing maximal matchings can thus be regarded as a black-box subroutine, which allows for the design of matching algorithms that are independent of the underlying computational model.

For example, in the *semi-streaming model* for processing large graphs [FKM+04, McG14], an algorithm makes few passes over the edges of the input graph in arbitrary order while maintaining a memory of size $O(n \operatorname{poly} \log n)$, where n is the number of vertices in the input graph. The Greedy matching algorithm, which inserts every arriving edge into an initially empty matching if possible, i.e., if none of its endpoints are already matched, yields a maximal matching and constitutes a one-pass semi-streaming algorithm. Since a maximal matching is at least half the size of a maximum matching, Greedy can also be regarded as a $\frac{1}{2}$-approximation semi-streaming algorithm for MM. While it is unknown whether it is possible to go beyond the approximation factor of $1/2$ in a single pass even on bipartite graphs (currently only approximation factors beyond $\frac{1}{1+\ln 2} \approx 0.59$ are ruled out [Kap21]), improved results are known

# Rounds	Vertex-query model ([KK20])	Edge-query model (this chapter)
1	$\frac{1}{2}$	$\frac{1}{2}$ (Theorem 4.15-1)
2	$\frac{1}{2}$	$\frac{1}{2} + \Theta(\frac{1}{n})$ (Theorem 4.15-2)
3	$\frac{3}{5} = 0.6$	$\frac{5}{8} = 0.625$ (Theorems 4.1 and 4.15-3)

Table 4.1: Optimal approximation ratios achievable for deterministic algorithms for MBM in the edge-query (this chapter) and vertex-query models ([KK20]).

for multiple passes, and, indeed, most multi-pass semi-streaming algorithms solely execute Greedy on carefully selected subgraphs in each pass (e.g. [EKMS12, AG13, Kon18b, ALT21]). This includes the state-of-the-art[1] $(1-\epsilon)$-approximation algorithm for MBM by Assadi et al. [ALT21], which executes Greedy $O(\frac{1}{\epsilon^2})$ times and thus runs in $O(\frac{1}{\epsilon^2})$ passes. This algorithm can easily be implemented in the Massively Parallel Computation model [KSV10] and also constitutes the state-of-the-art result in this model. As such, maximal matching query models capture these algorithms and allow for a systematic study of what can and cannot be achieved.

Our Results In this chapter, we give tight approximation ratios for deterministic edge-query algorithms for MBM for up to three rounds. Our results for one and two rounds as well as the lower bound for three rounds also hold for MM. In Table 4.1, we illustrate our bounds and compare them to the respective tight bounds that holds in the vertex-query model [KK20].

One Round. We show that the best approximation factor achievable in a single round for MBM is $\frac{1}{2}$, which matches the vertex-query setting (Theorem 4.15-1). Querying all potential edges, i.e., the query $V \times V$, yields a matching upper bound, even in general graphs.

Two Rounds. The approximation factor can be very slightly improved in two rounds, even in general graphs. Consider the algorithm that queries all edges $V \times V$ in the first round, which produces a maximal matching M_1 in the input graph. Next, pick any edge $uv \in M_1$ and query all edges incident to u and v with one endpoint unmatched by M_1. Then, we will either find a 3-augmenting path that allows us to augment the edge uv, or the edge uv is not 3-augmentable. In both cases, we establish that the resulting matching is a $\frac{1}{2} + \Theta(\frac{1}{n})$-approximation, and we also prove that no algorithm can do better, even in bipartite graphs (Theorem 4.15-2). While the $\Theta(\frac{1}{n})$ additive term is not significant in terms of an improved approximation guarantee, it nevertheless illustrates that the edge-query and vertex-query models behave slightly differently in the two rounds setting.

Three Rounds. As our main result, we give a deterministic 3-round algorithm for MBM in the edge-query model that produces a 0.625-approximation (Theorem 4.1), and we show that

[1] We note that the algorithm by [ALT21] yields a $(1-\epsilon)$-approximation in $O(\frac{1}{\epsilon^2})$ passes. Very recently, Assadi et al. [AJJ+22] gave a $(1-\epsilon)$-approximation algorithms for MBM that operates in $O(\frac{1}{\epsilon} \cdot \log n)$-passes, which, for very small values of ϵ, asymptotically requires fewer rounds than [ALT21].

this is best possible (Theorem 4.15-3). Our algorithm can be implemented in a straightforward way in the semi-streaming model and improves upon the previously best 3-pass semi-streaming 0.6111-approximation algorithm for MBM by Feldman and Szarf [FS22].

On 3-pass Semi-streaming Algorithms for MBM The first 3-pass semi-streaming algorithm for MBM was implicit in [FKM+04], explicitly mentioned in [KMM12], and analysed by Kale and Tirodkar [KT17], who showed that the approximation ratio is 0.6. Subsequently, binti Khalil and Konrad [KK20] proved that this algorithm constitutes an optimal 3-round vertex-query algorithm (and can thus also be implemented in the edge-query model). Various improvements have since been established via semi-streaming algorithms that cannot be implemented in the edge-query model. First, Esfandiari et al. [EHM16] gave a 0.605-approximation algorithm, which was then further improved by Konrad [Kon18b] who gave a randomized 0.6067-approximation algorithm. Very recently, Feldman and Szarf [FS22] gave a 0.6111-approximation. In this chapter, we improve the approximation factor to 0.625, again, with a 3-round deterministic edge-query algorithm.

While edge-query algorithms appear somewhat restricted in how they operate as compared to arbitrary semi-streaming algorithms, the literature illustrates that they are surprisingly powerful as they constitute the state-of-the-art algorithms in the 3-pass (this chapter) and $(1-\epsilon)$-approximation [ALT21] streaming settings for MBM.

Techniques We will first discuss the ideas behind our 3-round algorithm for MBM in the edge-query model, and then give the intuition behind our lower bound results.

3-round Query Algorithm. Our 3-round algorithm computes a maximal matching in the first round, and then finds augmenting paths in the subsequent rounds. This is a well-established technique, and almost all known 2-pass and 3-pass streaming algorithms operate in this fashion (e.g. [KMM12, EHM16, KT17, Kon18b, FS22]). To this end, denote by M_1 a maximal matching in the bipartite input graph $G = (A, B, E)$ that we obtain by querying all potential edges $A \times B$. We observe that every augmenting path for M_1 starts with an edge in $G_L = G[A(M_1) \cup \overline{B(M_1)}]$ and ends with an edge in $G_R = G[\overline{A(M_1)} \cup B(M_1)]$, where $A(M_1)$ denotes the A-vertices matched in M_1, and $\overline{A(M_1)} = A \setminus A(M_1)$ ($B(M_1)$ and $\overline{B(M_1)}$ are defined similarly). In our second round, we therefore compute maximal matchings M_L and M_R in G_L and G_R, respectively. Observe that we can indeed compute both of these matchings with the single query $\left(A(M_1) \times \overline{B(M_1)}\right) \cup \left(\overline{A(M_1)} \times B(M_1)\right)$ in the edge-query model. At this stage, we are guaranteed that the set $M_1 \cup M_L \cup M_R$ contains various length-2 paths consisting of one edge of M_1 and one additional edge either from M_L or M_R, and the usual idea employed in the literature is to complete these length-2 paths to length-3 augmenting paths using an additional pass over the data/an additional query. Indeed, if we attempted to complete the

length-2 paths by computing a maximal matching between the endpoints of length-2 paths in M_1 and the yet unmatched vertices in the third round then we obtain a 0.6-approximation.

Our key idea for the third query round that leads to an improvement over previous work is to simultaneously attempt to complete length-5 augmenting paths. To this end, denote by A' the endpoints of length-2 paths in $A(M_1)$, and by B' the endpoints of length-2 paths in $B(M_1)$. Our third round query consists of all potential edges interconnecting the vertices

$$A' \cup B' \cup \overline{A(M_1)} \cup \overline{B(M_1)} \ ,$$

i.e., we both attempt to complete length-2 paths to length-3 augmenting paths by considering the edges between A' and $\overline{B(M_1)}$ and between B' and $\overline{A(M_1)}$, but we also attempt to join two disjoint length-2 paths by connecting them via an edge between A' and B' to form a length-5 augmenting path. The main challenge in the analysis of this method is to address the complications that arise when bounding the size of the maximal matching returned from the queried subgraph in round 3.

Lower Bounds. The key idea behind our lower bound arguments is to keep track of the information revealed when the oracle returns a maximal matching M_i as a response to the query edges Q_i in round i. This approach was previously successfully employed by binti Khalil and Konrad [KK20] for obtaining optimal lower bounds in the vertex-query model. When a matching M_i is returned, the algorithm not only learns that the edges M_i are indeed contained in the input graph, but also that none of the edges of Q_i that connect vertices outside of $V(M_i)$ exist, which is due to the maximality of M_i in the subgraph $G[E \cap Q_i]$ of the input graph $G = (V, E)$. The main challenge lies in keeping track of the information revealed over the course of the algorithm while considering the complexity of *all* possible queries in each round. This is achieved by considering the vertex-induced subgraphs on carefully constructed partitions of the vertices while maintaining a *superset* of the information revealed to the algorithm in each part: We prove that, no matter the sequence of queries, the information about the edges that are guaranteed to exist and those that are guaranteed not to exist in each part of the partition is less than a certain superset of existing edges and non-existing edges that are easy to describe, up to isomorphism. Our arguments are substantially more involved than those for the vertex-query model [KK20], which is due to the fact that edge queries can have more complex structure.

Further Related Work The study of graph algorithms with query access to the input dates back to the works of Feige [Fei06] and Goldreich and Ron [GR08]. The literature distinguishes between *local queries* – such as vertex-degree queries, neighborhood queries, and edge-existence queries [Fei06, GR08, Beh21] – and *global queries* – such as (bipartite) independent set queries [BHR+20, AMM22], linear, or and cut queries [ACK21], and maximal matching queries as

studied in this chapter and [KK20]. We refer the reader to [AMM22] and the references therein for an overview.

Outline We first present our main result, a 3-round algorithm for MBM, in Section 4.2. Our lower bound results are discussed in Section 4.3, and we conclude with open questions in Section 4.4.

4.2 3-Round Algorithm

In this section, we present our 3-round query algorithm for MBM (see Algorithm 1). We prove the following result, which constitutes the main result of this chapter:

Theorem 4.1 (cf. Theorem 1)**.** *Algorithm 1 is a deterministic* 3*-round* $\frac{5}{8}$*-approximation algorithm for* MBM *in the edge-query model.*

Algorithm 1 Three Rounds using Maximal Matching Queries

Input: A bipartite graph $G = (A, B, E)$ and a maximal matching oracle QUERY
Output: A large matching M_{out} of G

First round

$M_1 \leftarrow \text{QUERY}(A \times B)$
$Q_L = \overline{A(M_1)} \times \overline{B(M_1)}$
$Q_R = \overline{A(M_1)} \times B(M_1)$

Second round

$M_2 \leftarrow \text{QUERY}(Q_L \cup Q_R)$
Let $A' \subseteq \overline{A(M_1)}$ and $B' \subseteq \overline{B(M_1)}$ be the endpoints of length-2 paths in $M_1 \cup M_2$
$Q' = (A' \times \overline{B(M_1)}) \cup (B' \times \overline{A(M_1)}) \cup (A' \times B')$

Third round

$M_3 \leftarrow \text{QUERY}(Q')$

Output

return M_{out}, the largest matching in $M_1 \cup M_2 \cup M_3$

Our algorithm operates on a bipartite input graph $G = (A, B, E)$, where only the vertex sets A and B are initially known to the algorithm. It only uses the edge-query maximal matching oracle to compute maximal matchings in subgraphs of G. Our algorithm initially computes a maximal matching M_1 of G in the first round. Then, in the second round, maximal matchings in the subgraphs $G_L = G[Q_L \cap E]$ and $G_R = G[Q_R \cap E]$ are computed (M_2 denotes the union of both matchings), where G_L consists of the edges connecting the B-vertices unmatched in M_1 to the A-vertices matched in M_1 and G_R is defined similarly with the roles of A and B reversed. The matching M_2 w.r.t. M_1 possibly finds some length-3 augmenting paths, which we denote

by P, while the remaining ones make up length-2 alternating paths. Last, in the third round, a maximal matching M_3 is computed in the subgraph induced by the vertices unmatched by M_1 and the endpoints of the length-2 paths in $M_1 \cup M_2$ that are also in $A(M_1) \cup B(M_1)$. Each edge of the matching M_3 thus completes length-3 and length-5 augmenting paths, which we denote by Q. See Figure 4.2 for an example run of the algorithm where $G_{\text{in}} = G[A(M_1) \cup B(M_1)]$.

Observation 4.2. The size of Q is exactly the size of M_3.

The goal of our analysis is to show that the size of the returned matching M_{out}, the largest matching in $M_1 \cup M_2 \cup M_3$, is always at least a $\frac{5}{8}$-approximation of a maximum matching M^*. To that end, we can always find a large maximal matching by appropriately augmenting M_1 with the augmenting paths $P \cup Q$. Although each edge in M_1 can be augmented by at most one augmenting path, the augmenting paths $P \cup Q$ are not necessarily vertex-disjoint since the vertices unmatched by M_1 may be incident to an edge in M_2 and also one in M_3. See Figure 4.2 for an example of this. However, we observe that the intersection multi-graph of the augmenting paths $P \cup Q$ has maximum degree 2 and, in particular, constitutes a collection of paths and even-length cycles. We can thus pick an independent set of non-overlapping augmenting paths of size $\frac{1}{2}(|P| + |Q|)$ and thus obtain the following:

$$|M_{\text{out}}| \geq |M_1| + \frac{1}{2}(|P| + |Q|). \tag{4.1}$$

We highlight here that either finding a large matching in the first round or finding many augmenting paths in the second round leaves fewer augmenting paths to be found in the third round. Therefore, the size of Q must be a decreasing function of $|M_1|$ and $|P|$. We subsequently formalise this by systematically bounding the quantities required to bound the size of M_3, which is equivalent to the size of Q (Observation 4.2). Theorem 4.1 then immediately follows from Equation (4.1).

Let M^* be a maximum matching in G with size $\mu(G)$. The first query finds a maximal matching M_1 of G, which is always at least half the size of M^* (Observation 4.3). By considering a maximum matching M^* such that $M^* \cup M_1$ contains no even-length alternating paths or cycles (Lemma 4.4), each vertex in $A(M_1)$ and $B(M_1)$ are endpoints of exactly one edge in M^* (Lemma 4.5) and we have that M_1 relates exactly to the the number of edges of M^* in G_L and G_R, respectively (Lemma 4.6). As such, for the remainder of the analysis, we assume this choice of M^*. See Figure 4.3 for an example of the implied structure.

Observation 4.3. $|M_1| = (\frac{1}{2} + \epsilon) \cdot \mu(G)$, $0 \leq \epsilon \leq \frac{1}{2}$.

Lemma 4.4. *There exists a maximum matching M^* such that $M^* \cup M_1$ has no even-length alternating paths or cycles.*

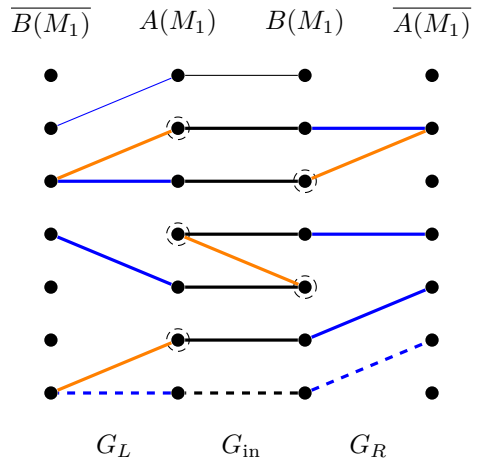


Figure 4.2: An example run of Algorithm 1 that showcases some possible intersections of the augmenting paths. The edges of M_1 are in black, the edges of M_2 are in blue, and the edges of M_3 are in orange. The dashed thick edges are those which belong to augmenting paths in P, whereas the solid thick edges are those which belong to Q. The vertices of A' and B' are circled.

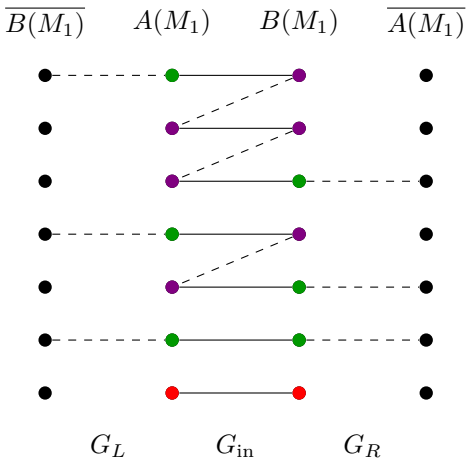


Figure 4.3: An example of the implied graph structure w.r.t. M^* and M_1. The edges of M_1 are the solid edges and the edges of $M^* \backslash M_1$ are dashed ones. The edges of $M^* \cap M_1$ are the solid edges not incident to any dashed ones. The red vertices represent A_* and B_*, the green ones represent A_{out} and B_{out}, and the violet ones represent A_{in} and B_{in}.

Proof. Let $P(M^*)$ be the set of even length alternating paths or cycles in $M^* \cup M_1$. If it is non-empty, we can use any $p \in P(M^*)$ to find another maximum matching $\hat{M}^*$.

Observe first that the edges of p alternate between edges in M_1 and M^* and is of even length. Therefore, we construct $\hat{M}^*$ from M^*, without decreasing its size, by replacing the edges of $M^* \cap p$ with the edges of $M_1 \cap p$. Finally, $\hat{M}^*$ is indeed a maximum matching since the edges of $M_1 \cap p$ are vertex disjoint from the edges of $M^* \backslash (M^* \cap p)$, otherwise p would not be an even length alternating path.

The only difference between the edges in $M^* \cup M_1$ and $\hat{M}^* \cup M_1$ are the removed edges $M^* \cap p$ which means that p is no longer an alternating path in $\hat{M}^* \cup M_1$ whereas all others remain unchanged. Therefore, $|P(\hat{M}^*)| = |P(M^*)| - 1$. Repeating this process until no such paths exist produces a maximum matching where the claim holds. □

Lemma 4.5. *Each vertex of $A(M_1)$ and $B(M_1)$ is the endpoint of an edge in M^*.*

Proof. Since M^* is a maximal matching, every edge of M_1 must be incident to at least one of its edges. By Lemma 4.4, there are no even length alternating paths in $M^* \cup M_1$; hence, every $(a, b) \in M_1$ is either an edge of M^* or part of an augmenting path in $M^* \cup M_1$. In either case, a and b are each endpoints of exactly one edge of M^*. □

Lemma 4.6. $|M^* \cap G_L| = |M^* \cap G_R| = (\frac{1}{2} - \epsilon) \cdot \mu(G)$.

Proof. Firstly, every edge of $M^* \oplus M_1$ is part of an alternating path or cycle since M_1 is maximal. Then, by Lemma 4.4, there are only odd length alternating paths, i.e., augmenting paths. Finally, any augmenting path must begin with an edge of M^* in G_L and end with one in G_R (or vice versa) with all other edges along the path belonging to G_{in}. See Figure 4.3 for an example. Thus, the number of edges of M^* in G_L and G_R, respectively, is the number of vertex-disjoint augmenting paths, which we subsequently show is exactly $(\frac{1}{2} - \epsilon) \cdot \mu(G)$ and thus implies the result.

Since M_1 and M^* are maximal matchings, every edge of M^* must be incident to at least one edge of M_1, and vice versa. In the case that $e \in M^* \cap M_1$, both these conditions are satisfied and e is an isolated edge in $M^* \cup M_1$ as both are matchings. Hence, in all other cases, the edges of M^* and M_1 belong to an alternating path in $M^* \oplus M_1$, all of which are necessarily vertex-disjoint since no two edges from a matching may share the same endpoint. Then, by Lemma 4.4, these are necessarily (odd length) augmenting paths. Finally, since M^* is a maximum matching of size $\mu(G)$ and each vertex-disjoint augmenting path increases the size of M_1 by 1, there must be exactly $\mu(G) - |M_1|$ many paths and the claim follows by Observation 4.3. □

We now have that the vertices $A(M_1)$ and $B(M_1)$ can be partitioned based on the kind of edge in M^* that they are endpoints of, i.e., by Lemma 4.5. Let $A_* := A(M^* \cap M_1)$, $A_{\text{out}} := A(M^* \cap G_L)$, and $A_{\text{in}} := A(M_1) \backslash (A_* \cup A_{\text{out}})$. Similarly define $B_* := B(M^* \cap M_1)$, $B_{\text{out}} := B(M^* \cap G_R)$ and $B_{\text{in}} := B(M_1) \backslash (B_* \cup B_{\text{out}})$. See the coloured vertices of Figure 4.3 for an example of these partitionings.

Lemma 4.7. $|A_{\text{in}}| = |B_{\text{in}}| = 2\epsilon \cdot \mu(G) - |M^* \cap M_1|$.

Proof. By construction of the partitions, we have that $|A_*| = |M^* \cap M_1| = |B_*|$, $|A_{\text{out}}| = |M^* \cap G_L|$ and $|B_{\text{out}}| = |M^* \cap G_R|$. Then, by Lemma 4.6, it follows that $|A_{\text{out}}| = |B_{\text{out}}| = (\frac{1}{2} - \epsilon) \cdot \mu(G)$. Finally, since $|A(M_1)| = |B(M_1)| = |M_1| = (\frac{1}{2} + \epsilon) \cdot \mu(G)$ by Observation 4.3, we have that $|A_{\text{in}}| = (\frac{1}{2} + \epsilon) \cdot \mu(G) - |A_{\text{out}}| - |A_*| = 2\epsilon \cdot \mu(G) - |M^* \cap M_1| = |B_{\text{in}}|$. □

Intuitively, we have that finding edges of M^* with the first query always puts us in a better situation. Therefore, we consider the effect that these edges have on the quantities that we bound. We introduce some helpful notation in this regard: For any matching M, we define $M(A_*)$ (and $M(B_*)$) as the edges of M that have one endpoint in A_* (resp. B_*), i.e., those which are incident to edges of $M^* \cap M_1$.

The second query finds a maximal matching M_2, which is the union of (vertex-disjoint) maximal matchings M_L in G_L and M_R in G_R. Each edge of M_2 forms the beginning of an

alternating path in $M_1 \cup M_2$, some of which may immediately form length-3 augmenting paths P while the rest form length-2 alternating paths P', which are extended in the third round. Note that the endpoints of P' that are in $A(M_1)$ and $B(M_1)$ are the vertex sets A' and B', respectively. We then partition A' and B' such that $A'_{\text{out}} := A' \cap A_{\text{out}}$ and similarly define A'_{in}, A'_*, B'_{out}, B'_{in}, and B'_*.

Lemma 4.8. $|M^* \cap M_1| \geq \frac{1}{2} \cdot (|M_L(A_*)| + |M_R(B_*)|)$.

Proof. By definition, every edge of the matchings $M_L(A_*)$ and $M_R(B_*)$ is incident to an edge of $M^* \cap M_1$. Hence, $|M^* \cap M_1| \geq |M_L(A_*)|$ and $|M^* \cap M_1| \geq |M_R(B_*)|$, which implies the result. □

Lemma 4.9. $|M_2| \geq (\frac{1}{2} - \epsilon) \cdot \mu(G) + \frac{1}{2}(|M_L(A_*)| + |M_R(B_*)|)$

Proof. Consider the edges of $M^* \cap G_L$. Every edge of M_L is incident to at most two edges of $M^* \cap G_L$, and every edge of $M_L(A_*)$ is incident to at most one of the edges of $M^* \cap G_L$. By a counting argument, we have that $|M_L| \geq \frac{1}{2}(|M^* \cap G_L| - |M_L(A_*)|) + |M_L(A_*)|$. Then, by Lemma 4.6, it follows that $|M_L| \geq \frac{1}{2}\left((\frac{1}{2} - \epsilon) \cdot \mu(G) + |M_L(A_*)|\right)$. We similarly bound $|M_R|$ w.r.t. $M_R(B_*)$ and obtain the results since $|M_2| = |M_L| + |M_R|$. □

Lemma 4.10. $|P'| = |A'_{\text{in}}| + |B'_{\text{in}}| + |A'_{\text{out}}| + |B'_{\text{out}}| + |A'_*| + |B'_*| = |M_2| - 2|P|$.

Proof. Each length-2 alternating path in P' has an endpoint in either $A(M_1)$ or $B(M_1)$, but not both; thus, we have that $|P'| = |A'| + |B'|$ and the first equality follows by definition of the partitions of A' and B'. For the subsequent equality, we have that every edge of M_L contributes to a length-2 alternating path except for the ones which contribute to length-3 augmenting paths. This gives $|M_L| - |P|$ of them which have an edge in M_L. A similar reasoning w.r.t. M_R shows that $|M_R| - |P|$ of them have an edge in M_R. The result then follows since the paths in P' either have an edge in M_L or in M_R, but never both. □

Lemma 4.11. $|A'_*| \leq |M_R(B_*)|$ *and* $|B'_*| \leq |M_L(A_*)|$.

Proof. Consider any vertex in $a \in A'_*$. By definition, a is the endpoint of an edge $(a, b) \in M^* \cap M_1$, which implies that $b \in B_*$. Since a is the endpoint of a path $p \in P'$, we have that $p = (a, b, a_R)$ where (a_R, b) must be an edge of $M_R(B_*)$. Finally, every $a \in A'_*$ has a unique $(a_R, b) \in M_R(B_*)$ since each path in P' is vertex-disjoint and thus the first inequality follows. The second inequality follows similarly w.r.t. B_* and $M_L(A_*)$ instead. □

The third query finds a maximal matching M_3 in the graph $G' = G[Q' \cap E]$, which implies that the size of M_3 is at least half of $\mu(G')$. As such, it is sufficient to bound $\mu(G')$. To that end, we bound the number of edges of M^* in G', which we accomplish by decomposing G'

into edge-disjoint subgraphs $G'_L = G_L \cap G'$, $G'_R = G_R \cap G'$, and $G'_{\text{in}} = G_{\text{in}} \cap G'$. Using the quantities we have previously bounded, we then obtain our final bound on the size of M_3 as a decreasing function of $|M_1|$, in terms of ϵ, and $|P|$.

Lemma 4.12. *$|M^* \cap G'_L| = |A'_{\text{out}}|$, $|M^* \cap G'_R| = |B'_{\text{out}}|$, and $|M^* \cap G'_{\text{in}}| \geq |A'_{\text{in}}| + |B'_{\text{in}}| - |A_{\text{in}}|$.*

Proof. By definition, every vertex of A'_{out} is incident to an edge in $M^* \cap G'_L$, and vice versa; hence, it holds that $|M^* \cap G'_L| = |A'_{\text{out}}|$. A similar argument shows that $|M^* \cap G'_R| = |B'_{\text{out}}|$ holds. To bound $|M^* \cap G'_{\text{in}}|$, consider an edge $(a, b) \in (M^* \setminus M_1) \cap G_{\text{in}}$. By definition, $a \in A_{\text{in}}$ and $b \in B_{\text{in}}$; however, $(a, b) \in M^* \cap G'_{\text{in}}$ if and only if $a \in A'_{\text{in}}$ and $b \in B'_{\text{in}}$. Thus, there are at most $(|A_{\text{in}}| - |A'_{\text{in}}|) + (|B_{\text{in}}| - |B'_{\text{in}}|)$ edges of $(M^* \setminus M_1) \cap G_{\text{in}}$ which are not in $M^* \cap G'_{\text{in}}$. By definition, it holds that $|(M^* \setminus M_1) \cap G_{\text{in}}| = |B_{\text{in}}|$ and it follows that $|M^* \cap G'_{\text{in}}| \geq |A'_{\text{in}}| + |B'_{\text{in}}| - |A_{\text{in}}|$. □

Lemma 4.13. $|M_3| \geq (\frac{1}{4} - \frac{3\epsilon}{2}) \cdot \mu(G) - |P|$.

Proof. By rearranging the equation in Lemma 4.10 and applying Lemma 4.11, we have that

$$|A'_{\text{in}}| + |B'_{\text{in}}| + |A'_{\text{out}}| + |B'_{\text{out}}| \geq |M_2| - 2|P| - |M_R(B_*)| - |M_L(A_*)|. \tag{4.2}$$

We subsequently bound $\mu(G')$, which the result follows from since $|M_3| \geq \frac{1}{2} \cdot \mu(G')$.

$$\begin{aligned}
\mu(G') &\geq |M^* \cap G'_L| + |M^* \cap G'_R| + |M^* \cap G'_{\text{in}}| && \text{(edge-disjoint subgraphs)} \\
&\geq |A'_{\text{out}}| + |B'_{\text{out}}| + |A'_{\text{in}}| + |B'_{\text{in}}| - |A_{\text{in}}| && \text{(by Lemma 4.12)} \\
&\geq |M_2| - 2|P| - |M_L(A_*)| - |M_R(B_*)| - |A_{\text{in}}| && \text{(by Equation (4.2))} \\
&\geq (\frac{1}{2} - \epsilon) \cdot \mu(G) - \frac{1}{2}(|M_L(A_*)| + |M_R(B_*)|) - 2|P| - |A_{\text{in}}| && \text{(by Lemma 4.9)} \\
&\geq (\frac{1}{2} - \epsilon) \cdot \mu(G) - |M^* \cap M_1| - 2|P| - |A_{\text{in}}| && \text{(by Lemma 4.8)} \\
&= (\frac{1}{2} - \epsilon) \cdot \mu(G) - |M^* \cap M_1| - 2|P| - (2\epsilon \cdot \mu(G) - |M^* \cap M_1|) && \text{(by Lemma 4.7)} \\
&= (\frac{1}{2} - 3\epsilon) \cdot \mu(G) - 2|P|. && \square
\end{aligned}$$

We are now ready to bound the size of the returned matching M_{out} w.r.t. $\mu(G)$, the size of the maximum matching M^*, thus proving Theorem 4.1.

Lemma 4.14. *The large matching M_{out} returned by Algorithm 1 is always at least a $\frac{5}{8}$-approximation of the size of a maximum matching in the input graph G.*

Proof.

$$\begin{aligned}
|M_{\text{out}}| &\geq |M_1| + \frac{1}{2}(|P| + |Q|) && \text{(by Equation (4.1))} \\
&= (\frac{1}{2} + \epsilon) \cdot \mu(G) + \frac{1}{2}(|P| + |M_3|) && \text{(by Observations 4.2 and 4.3)}
\end{aligned}$$

$$\geq (\frac{1}{2}+\epsilon)\cdot\mu(G) + \frac{1}{2}(|P| + (\frac{1}{4}-\frac{3\epsilon}{2})\cdot\mu(G) - |P|) \qquad \text{(by Lemma 4.13)}$$
$$= (\frac{5}{8}+\frac{\epsilon}{4})\cdot\mu(G). \qquad \square$$

Finally, we show in Section 4.2.1 that the analysis of the approximation factor of Algorithm 1 is tight up to additive $\Theta(\frac{1}{n})$ factors, even when $|M_1| > \frac{1}{2}\cdot\mu(G)$, i.e., when $\epsilon > 0$.

4.2.1 Hard Instance for our 3-Round Algorithm

In this subsection, we construct a n-vertex bipartite graph $G = (A, B, E)$ where $|A| = |B| = \frac{n}{2}$ that is a hard instance for our 3-round algorithm, Algorithm 1. In particular, we show that the analysis of the approximation factor of our algorithm is tight up to (additive) $\Theta(\frac{1}{n})$ factors, even when the first round maximal matching is up to a $\frac{2}{3}$-approximation – any better than this is trivially strictly better than what is achieved by the analysis.

Towards constructing a hard instance G, let M^* be any perfect matching on the vertices $A \cup B$ and let M_1, M_2, M_3 be the matchings learned in each round of the algorithm. These matchings ultimately make up the edge set of the hard instance $G = (A, B, M^* \cup M_1 \cup M_2 \cup M_3)$ where $\mu(G) = \frac{n}{2}$. More specifically, for each matching learned, we argue that its size satisfies the guarantees of the analysis with equality up to (additive) constant factors. Then, we argue that the size of the largest matching in $M_1 \cup M_2 \cup M_3$, i.e., the matching outputted by the algorithm, also satisfies the analysis with equality up to constant factors. Therefore, the approximation factor of the matching outputted by Algorithm 1 satisfies the analysis in Section 4.2 up to $\Theta(\frac{1}{n})$ factors, as required. See Figure 4.4 for an illustration.

First Round. The algorithm queries the complete graph $G[A \cup B]$. We construct M_1 to be a matching of size $(\frac{1}{2}+\epsilon)\cdot\frac{n}{2}$ for some fixed choice of $0 \leq \epsilon \leq \frac{1}{6}$ such that $M^* \cup M_1$ has $(\frac{1}{2}-3\epsilon)\cdot\frac{n}{2}$ many length-3 augmenting paths and $2\epsilon\cdot\frac{n}{2}$ many length-5 augmenting paths[2]. Therefore, M_1 is maximal w.r.t. M^* and satisfies Observation 4.3. Note that we will not use any edges of $\overline{A(M_1)} \times \overline{B(M_1)}$ to construct the matchings M_2 and M_3, and thus M_1 is maximal in G, i.e., the initial query.

Second Round. The algorithm queries a union of the vertex-induced subgraphs $G_L = G[A(M_1) \cup \overline{B(M_1)}]$ and $G_R = G[\overline{A(M_1)} \cup B(M_1)]$. As such, no edges of M_1 are contained in $G_L \cup G_R$ and thus any maximal matching only needs to be maximal among the edges $M^* \cap (G_L \cup G_R)$ and $M_3 \cap (G_L \cup G_R)$.

To consider $M^* \cap (G_L \cup G_R)$, observe first that each augmenting path p in $M^* \cup M_1$ has exactly one edge l_p^* in $M^* \cap G_L$ and exactly one edge r_p^* in $M^* \cap G_R$. Then, we pair together

[2] As long as ε is chosen such that $(\frac{1}{2}+\epsilon)\cdot\frac{n}{2}$ is an integer, then the number of length-3 and length-5 augmenting paths will also be integers.

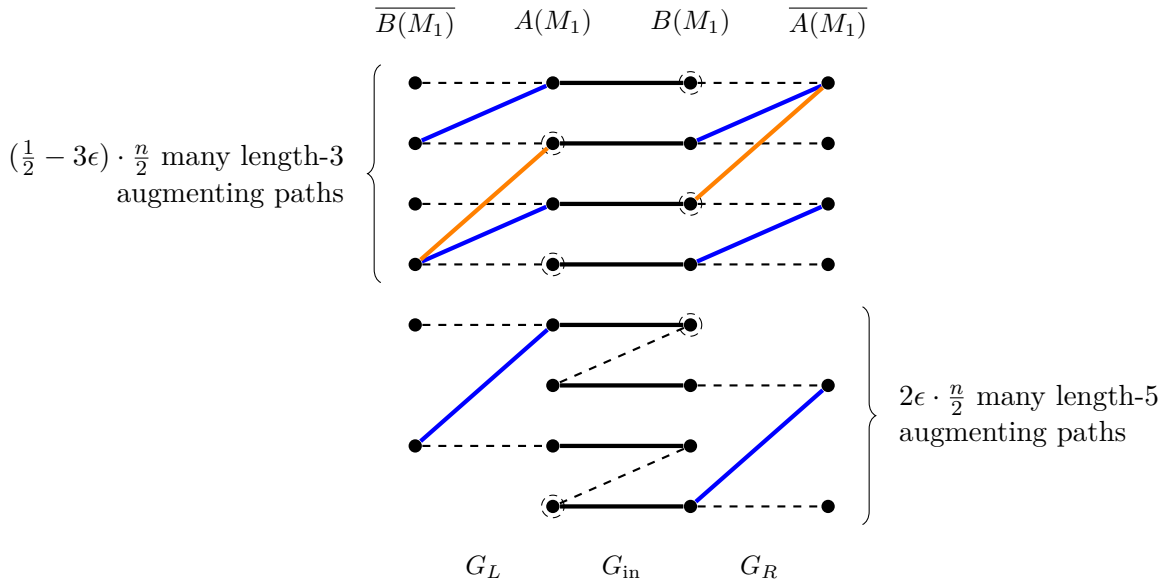


Figure 4.4: An illustration of the hard instance n-vertex graph G. The dashed black edges are M^*. The solid black edges are M_1, the blue ones are M_2, and the orange ones are M_3. The vertices of A' and B' are circled.

length-3 paths and, similarly, length-5 paths. Now, for each pair of augmenting paths p, q, we add the edges $\left(A(l_p^*), B(l_q^*)\right)$ and $\left(A(r_p^*), B(r_q^*)\right)$ to an initially empty matching M_2. In the case where there is an odd number of length-3 (or length-5) augmenting paths, exactly one path p_3' (resp. p_5') does not belong to a pair; hence, its $M^* \cap G_L$ and $M^* \cap G_R$ edges are not incident to M_2. In either case, we simply add the respective edges to M_2, ensuring its maximality among the edges $M^* \cap (G_L \cup G_R)$. To consider $M_3 \cap (G_L \cup G_R)$, we will simply avoid the use of any of the edges $\overline{A(M_2)} \times \overline{B(M_2)}$ that are also in $G_L \cup G_R$ when constructing M_3, and thus M_2 is maximal in $G_L \cup G_R$.

It follows that two edges are added to M_2 for every pair of augmenting paths and that two edges are added to M_2 for each of the paths p_3' and p_5' if they exist. Overall, this implies that

$$\begin{aligned}|M_2| &= 2 \cdot \frac{1}{2}\left((\frac{1}{2} - 3\epsilon) \cdot \frac{n}{2} - \mathbb{I}(p_3')\right) + 2 \cdot \frac{1}{2}\left(2\epsilon \cdot \frac{n}{2} - \mathbb{I}(p_5')\right) + 2 \cdot \mathbb{I}(p_3') + 2 \cdot \mathbb{I}(p_5') \\ &= (\frac{1}{2} - \epsilon) \cdot \frac{n}{2} + \mathbb{I}(p_3') + \mathbb{I}(p_5')\end{aligned}$$

where $\mathbb{I}(p) = 1$ in the case that p exists and $\mathbb{I}(p) = 0$ otherwise, that is, it represents the indicator variable for whether there is an odd number of length-3 or length-5 augmenting paths, respectively. This satisfies Lemma 4.9 with equality up to constant factors.

Third Round. The algorithm queries the vertex-induced subgraph $G' = G[A' \cup B' \cup \overline{A(M_1)} \cup \overline{B(M_1)}]$ where A' and B' are the endpoints of length-2 paths in $M_1 \cup M_2$ that are also contained

in $A(M_1) \cup B(M_1)$. As such, no edges of M_1 or M_2 are contained in G' and thus any maximal matching only needs to be maximal among the edges $M^* \cap G'$.

Observe that by construction of M_2, the pairs of length-5 augmenting paths in $M^* \cap M_1$ (described previously) do not have any edges of M^* that are in G'. However, in the case where the path p'_5 exists, i.e., there are an odd number of length-5 augmenting paths, its edge in $M^* \cap G_{\text{in}}$ is in G', which we add to an initially empty matching M_3. Note that in the case where the path p'_3 exists, it does not have any edges in G' – although this augmenting path is already contained in $M_1 \cup M_2$.

It remains to consider the edges that belong to pairs of length-3 augmenting paths. Each pair p, q is such that only the edges $l^*_q \in M^* \cap G_L$ and $r^*_p \in M^* \cap G_R$ are in G'. As such, we group together the pairs into pairs of pairs $(p_1, q_1), (p_2, q_2)$ such that the edges $l^*_{q_1}, l^*_{q_2} \in M^* \cap G_L$ and $r^*_{p_1}, r^*_{p_2} \in M^* \cap G_R$ are in G'. Then, for every pair of pairs, we add the edges $\left(A(l^*_{q_1}), B(l^*_{q_2})\right)$ and $\left(A(r^*_{p_1}), B(r^*_{p_2})\right)$ to M_3. In the case where there are an odd number of pairs, there exists a pair p'', q'' whose edges $l^*_{q''} \in M^* \cap G_L$ and $r^*_{p''} \in M^* \cap G_R$ are in G' and not yet incident to any edge in M_3; hence, we add exactly these edges to M_3 to ensure that it is maximal in G'.

It follows that one edge is added to M_3 if p'_5 exists, two edges are added to M_3 for every pair of pairs, and two edges are added to M_3 if the pair p'', q'' exists. Overall, this implies that

$$\begin{aligned}|M_3| &= \mathbb{I}(p'_5) + 2 \cdot \frac{1}{2} \cdot \left(\frac{1}{2} \cdot \left((\frac{1}{2} - 3\epsilon) \cdot \frac{n}{2} - \mathbb{I}(p'_3)\right) - \mathbb{I}(p'', q'')\right) + 2 \cdot \mathbb{I}(p'', q'') \\ &= (\frac{1}{4} - \frac{3\epsilon}{2}) \cdot \frac{n}{2} - \frac{\mathbb{I}(p'_3)}{2} + \mathbb{I}(p'', q'') + \mathbb{I}(p'_5).\end{aligned}$$

This satisfies Lemma 4.13 with equality up to constant factors since $|P| = \mathbb{I}(p'_3)$ where P is defined (in Lemma 4.13) to be the number of length-3 augmenting paths in $M_1 \cup M_2$.

Outputted Matching. The algorithm outputs the largest matching M_{out} among the learned edges $M_1 \cup M_2 \cup M_3$. These learned edges are structured as vertex-disjoint gadgets that we consider separately as follows:

- When p'_3 and/or p'_5 exist, all edges of the augmenting paths p'_3 and/or p'_5 are among the matching edges and thus they contribute two and three edges to M_{out}, respectively.
- When the pair p'', q'' of length-3 augmenting paths exists, their learned edges form exactly a single 6-cycle and thus contributes three edges to M_{out}.
- Each pair of length-5 augmenting paths contains four edges of M_1, two edges of M_2, and no edges from M_3; however, both edges of M_2 are incident to distinct edges of M_1, forming length-2 paths. As such, each pair contributes four edges to M_{out}. See Figure 4.4 for an illustration.
- Each pair of pairs of length-3 augmenting paths contains four edges of M_1, four edges of M_2, and two edges from M_3. The two edges of M_3, along with two edges each of M_1

and M_2, form a 6-cycle. On the other hand, the remaining two edges of M_1 and M_2 form two length-2 paths. Therefore, each pair of pairs contributes five edges to M_{out}. See Figure 4.4 for an illustration.

This considers all the learned edges and, overall, implies that

$$\begin{aligned}|M_{\text{out}}| &= 2 \cdot \mathbb{I}(p_3') + 3 \cdot \mathbb{I}(p_5') + 3 \cdot \mathbb{I}(p'', q'') + 4 \cdot \frac{1}{2} \cdot \left(2\epsilon \cdot \frac{n}{2} - \mathbb{I}(p_5')\right) \\ &\quad + 5 \cdot \frac{1}{2} \cdot \left(\frac{1}{2} \cdot \left((\frac{1}{2} - 3\epsilon) \cdot \frac{n}{2} - \mathbb{I}(p_3')\right) - \mathbb{I}(p'', q'')\right) \\ &= \left(\frac{5}{8} + \frac{\epsilon}{4}\right) \cdot \frac{n}{2} + \underbrace{\frac{3}{4} \cdot \mathbb{I}(p_3') + \mathbb{I}(p_5') + \frac{1}{2} \cdot \mathbb{I}(p'', q'')}_{=O(1)}.\end{aligned}$$

This satisfies Lemma 4.14 with equality up to constant factors.

Finally, the approximation factor of the hard instance G is then $\frac{|M_{\text{out}}|}{\mu(G)} = \frac{5}{8} + \frac{\epsilon}{4} + \frac{c}{n}$ where $0 \leq c \leq \frac{9}{2}$ is dependent on the choice of ε. This satisfies the analysis with equality up to the $\Theta(\frac{1}{n})$ factor.

4.3 Lower Bound Results

In this section, we give our lower bounds for edge-query algorithms for MM for up to 3 rounds, showing that our 3-round algorithm is best possible:

Theorem 4.15 (cf. Theorem 2)**.** *There does* ***not*** *exist a deterministic algorithm on a n-vertex input graph for MM in the maximal matching edge-query model that achieves a better than*

1. $\frac{1}{2}$*-approximation in* 1 *round,*
2. $\left(\frac{1}{2} + \frac{2}{n}\right)$*-approximation in* 2 *rounds, and*
3. $\left(\frac{5}{8} + \frac{12}{n}\right)$*-approximation in* 3 *rounds.*

We prove our lower bounds by considering a game between a player, i.e., the algorithm, and an oracle in the edge-query model. The goal of the player is to learn a large matching in the underlying bipartite graph $G = (A, B, E)$ that is adversarially constructed by the oracle along the way. The player initially only knows the vertices A and B and is allowed to query the oracle with any set of edges $Q \subseteq A \times B$ in each round, typically basing the query on any information about G revealed in previous rounds. The oracle then returns an adversarially chosen maximal matching in the subgraph $G[E \cap Q]$, revealing as little information about a large matching as possible. Throughout the game, once information about G is revealed, it may not be altered in subsequent rounds.

Let Q_i be the player's query and let M_i be the maximal matching returned by the oracle in round i. The player learns that the edges M_i are present in G and that the edges in Q_i

with both endpoints unmatched by M_i do not exist in G. The player thus learns about both *edges* and *non-edges*. As such, we use *structure graphs* to encapsulate the information known by the player up to graph isomorphisms, providing a simple representation in which to prove our lower bounds – similar to the work by binti Khalil and Konrad [KK20].

Definition 4.16 (Structure Graph [KK20])**.** A 4-tuple (A, B, E, F) is a *bipartite structure graph* if E and F are disjoint sets of edges such that (A, B, E) and (A, B, F) are bipartite graphs. The set E corresponds to the set of edges learnt by the algorithm, and the set F corresponds to the set of non-edges learnt.

A player always begins with the empty structure graph $H_0 = (A, B, E_0, F_0)$ where $E_0 = F_0 = \emptyset$. In a game of r rounds, the structure graphs $H_1, H_2, \ldots, H_r$ represent the information (edges and non-edges) learned by the player after each round, which are based purely on the player's queries $Q_1, Q_2, \ldots, Q_r$ and the oracle's adversarially returned matchings $M_1, M_2, \ldots, M_r$. Consider a player's structure graph H_i w.r.t. H_{i-1} for any $i \in [r]$. It consists of the edges $E_i = E_{i-1} \cup M_i$ and the non-edges $F_i = F_{i-1} \cup N_i$ where $N_i = Q_i \cap (\overline{A(M_i)} \times \overline{B(M_i)})$ ensures that M_i is maximal. The player's information at the end of a round is then necessarily a superset of the information known at the end of the previous round, i.e., $E_i \supseteq E_{i-1}$ and $F_i \supseteq F_{i-1}$. Hence, the structure graph H_i *dominates* H_{i-1}, which we say in general for any structure graph that is a superset of the information of another up to graph isomorphism. The underlying graph G may then be any graph that is *consistent* with *all* the information H_r revealed to the player by the end of round r.

Definition 4.17 (Consistent)**.** Let $G = (A, B, E)$ be any bipartite graph and let $H_i = (A, B, E_i, F_i)$ be a bipartite structure graph. G is *consistent* with H_i iff $E \supseteq E_i$ and $E \cap F_i = \emptyset$.

The largest matching a player who knows H_r may output is the maximum matching M_r^{out} in (A, B, E_r). Therefore, the oracle adversarially constructs the graph G so that G is consistent with H_r and so that G has the largest possible maximum matching. This implies that the approximation factor of H_r is $\frac{|M_r^{\text{out}}|}{\mu(G)}$. Note that H_r is strongly dependent on the player's sequence of queries $Q_1, Q_2, ..., Q_r$. Altering even a single query could alter H_r, the player's largest matching M_r^{out} and, most importantly, the approximation factor of H_r. Hence, the goal for proving our lower bound results is to find an upper bound on the approximation factor achieved by any sequence of queries $Q_1, \ldots, Q_r$ for $r = 1, 2$ and 3.

Before proceeding with our analysis, we first present the ideas that we employ to prove our lower bounds. To generally consider all possible queries in each round $i \in [r]$, we allow the oracle to commit to *more* information than is revealed to the player, denoted by the structure graph $\tilde{H}_i$. In particular, we show that $\tilde{H}_i$ dominates the player's structure graph H_i learned regardless of the query Q_i. Then, at the end of round i, the player is assumed to have knowledge of the oracle's structure graph $\tilde{H}_i$. This implies that $\tilde{H}_r$ dominates the structure graph learned

by the player for any sequence of queries $Q_1, \ldots, Q_r$. We also allow the oracle to partition the vertices of the graph and consider the vertex-induced subgraph of each part independently. By making the partition a function of the query, we create desirable properties in each part. This, however, does not consider the edges that cross the partition, which are thus asserted as non-edges. Formally, we recombine the structure graphs learned in each part using the *disjoint union* (Definition 4.18) at the end of round r. Then, the approximation factor of the recombined structure graph follows naturally from the independent parts by Observations 4.19 and 4.20.

Definition 4.18 (Disjoint Union)**.** Let (A_x, B_x) and (A_y, B_y) represent an arbitrary partitioning of the vertices A and B into two parts and let $H_x = (A_x, B_x, E_x, F_x)$ and $H_y = (A_y, B_y, E_y, F_y)$ be any bipartite structure graphs. Then, their disjoint union is $H_x \dot\cup H_y = (A, B, E_x \cup E_y, F_x \cup F_y \cup (A_x \times B_y) \cup (A_y \times B_x))$.

Observation 4.19. Let H_x and H_y be bipartite structure graphs on disjoint sets of vertices with largest output matchings M_x^{out} and M_y^{out}, respectively. Then, the largest output matching of $H_x \dot\cup H_y$ is of size $|M_x^{\text{out}}| + |M_y^{\text{out}}|$.

Observation 4.20. Let H_x and H_y be bipartite structure graphs on disjoint sets of vertices with consistent graphs G_x and G_y, respectively. Then, there exists a graph G consistent with $H_x \dot\cup H_y$ such that $\mu(G) = \mu(G_x) + \mu(G_y)$.

We begin our analysis with the following simplifying assumption:

Assumption 4.21. In each round $1 \leq i \leq r$, we assume that the query Q_i does not contain any edges or non-edges already learned by the player.

Reason. Let $H = (A, B, E, F)$ be the structure graph known by the player. Let $e \in E$ and $f \in F$. If $e \in Q_i$, then the oracle can add e to the returned matching M_i without revealing any information about the edges incident to e that the player could have otherwise learned; thus, the query $Q_i \backslash \{e\}$ could never reveal less information than Q_i. If $f \in Q_i$, then f would never be in M_i; hence, $Q_i \backslash \{f\}$ would be an equivalent query. □

Let $A = A_{\text{in}} \cup A_{\text{out}}$ and $B = B_{\text{in}} \cup B_{\text{out}}$ be such that A_{in}, A_{out}, B_{in} and B_{out} are disjoint sets of vertices of size $\frac{n}{4}$ where n is the number of vertices and a multiple of 4. We further assert that $\frac{n}{4}$ is odd. Then, the player begins with only the knowledge of A and B, i.e., the empty structure graph H_0.

4.3.1 First Round

Let $\tilde{M}_1$ be a matching of size $\frac{n}{4}$ that matches A_{in} to B_{in}. We assert its maximality by letting $\tilde{N}_1^{\max} = A_{\text{out}} \times B_{\text{out}}$ be non-edges. Additionally, the non-edges $\tilde{N}_1^{\text{ind}} = (A_{\text{in}} \times B_{\text{in}}) \backslash \tilde{M}_1$ assert

that it is an induced matching[3]. Then, we define $\tilde{H}_1 = (A, B, \tilde{E}_1, \tilde{F}_1)$ where $\tilde{E}_1 = \tilde{M}_1$ and $\tilde{F}_1 = \tilde{N}_1 = \tilde{N}_1^{\max} \cup \tilde{N}_1^{\text{ind}}$. See Figure 4.5 for an illustration.

Remark. $\tilde{H}_1$ can be defined on any subset of vertices $A' \subseteq A$ and $B' \subseteq B$ such that $|A'_{\text{in}}| = |B'_{\text{in}}| = |A'_{\text{out}}| = |B'_{\text{out}}|$. We later use such generalisations, denoted as $\tilde{H}_1(A', B')$.

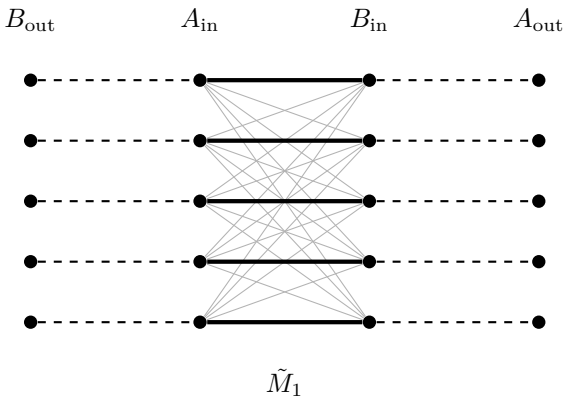


Figure 4.5: An illustration of structure graph $\tilde{H}_1$. The thick solid black edges represent the matching $\tilde{M}_1$. The non-edges $\tilde{N}_1^{\max}$ are implicit by the layout of the vertices and the non-edges $\tilde{N}_1^{\text{ind}}$ are the grey edges. The dashed black edges are a perfect matching in a worst-case underlying graph.

Lemma 4.22. *Any structure graph H_1 learned by the player is dominated by $\tilde{H}_1$.*

Proof. Let Q_1 be any arbitrary query and let $M^*(Q_1)$ be a maximum matching in the query graph (A, B, Q_1). If $|M^*(Q_1)| \geq |\tilde{M}_1|$, then let $M_1 \subseteq M^*(Q_1)$ be a subset of size $|\tilde{M}_1|$. We assert the maximality of matching M_1 with the non-edges $N_1 = \overline{A(M_1)} \times \overline{B(M_1)}$. Otherwise, $|M^*(Q_1)| < |\tilde{M}_1|$ and we let $M_1 = M^*(Q_1)$, which is trivially a maximal matching among the edges of the query Q_1; hence, $N_1 = \emptyset$.

Finally, in either case, let σ be a graph isomorphism such that $M_1 \subseteq \sigma(\tilde{M}_1)$, which implies that $N_1 \subseteq \sigma(\tilde{N}_1)$. Therefore, the player's H_1 learned is always dominated by $\tilde{H}_1$. □

Lemma 4.23. *The approximation factor of the structure graph $\tilde{H}_1$ is $\frac{1}{2}$.*

Proof. The largest matching $\tilde{M}_1^{\text{out}}$ that the player who knows $\tilde{H}_1$ may output is the matching $\tilde{M}_1$, which matches half of the vertices. Since no information regarding the edges in $A_{\text{in}} \times B_{\text{out}}$ or $A_{\text{out}} \times B_{\text{in}}$ has been revealed, we can choose a graph G consistent with $\tilde{H}_1$ that has a perfect matching, which is of size $2 \cdot |\tilde{M}_1|$, thus proving the result. In particular, it has a matching that arbitrarily matches A_{in} to B_{out} and A_{out} to B_{in} (see Figure 4.5). □

[3]Committing to an induced matching simplifies the arguments in the subsequent rounds without affecting the approximation factor of the player's structure graphs.

Lemma 4.22 shows that $\tilde{H}_1$ dominates all possible structure graphs learned by the player by the end of round 1, i.e., after query Q_1. Thus, Lemma 4.23 immediately implies Theorem 4.15-1.

4.3.2 Second Round

Let $\tilde{M}_L \subseteq (A_{\text{in}} \times B_{\text{out}})$ and $\tilde{M}_R \subseteq (A_{\text{out}} \times B_{\text{in}})$ be matchings of size $\left\lfloor \frac{|\tilde{M}_1|}{2} \right\rfloor$ such that $\tilde{M}_1 \cup \tilde{M}_L \cup \tilde{M}_R$ has no length-3 paths. Let $\tilde{N}_L^{\max} = A_{\text{in}} \setminus A(\tilde{M}_L) \times B_{\text{out}} \setminus B(\tilde{M}_L)$ and $\tilde{N}_R^{\max} = A_{\text{out}} \setminus A(\tilde{M}_R) \times B_{\text{in}} \setminus B(\tilde{M}_R)$ be the non-edges that assert the maximality of the matching $\tilde{M}_2 = \tilde{M}_L \cup \tilde{M}_R$ among the unknown edges. Let $\tilde{N}_L^{\text{ind}} = \left(A(\tilde{M}_L) \times B(\tilde{M}_L)\right) \setminus \tilde{M}_L$ and $\tilde{N}_R^{\text{ind}} = \left(A(\tilde{M}_R) \times B(\tilde{M}_R)\right) \setminus \tilde{M}_R$ be the non-edges required to make the matching induced. Additionally, if $|\tilde{M}_1|$ is odd, then let $e_{\text{in}}^* \in \tilde{M}_1$ be the only edge with both endpoints unmatched by $\tilde{M}_L$ and $\tilde{M}_R$, and similarly let $a_{\text{out}} \in A_{\text{out}}$ and $b_{\text{out}} \in B_{\text{out}}$ be any vertices unmatched by $\tilde{M}_L$ and $\tilde{M}_R$. We assert that e_{in}^* is an isolated edge and that a_{out} and b_{out} are isolated vertices[4], which implies the non-edges $\tilde{N}_* = (A(e_{\text{in}}^*) \times B \setminus B(e_{\text{in}}^*)) \cup (A \setminus A(e_{\text{in}}^*) \times B(e_{\text{in}}^*)) \cup (A \times \{b_{\text{out}}\}) \cup (\{a_{\text{out}}\} \times B)$. Otherwise, if $|\tilde{M}_1|$ is even, we let $\tilde{N}_* = \emptyset$. Then, we define $\tilde{H}_2 = (A, B, \tilde{E}_1 \cup \tilde{M}_2, \tilde{F}_1 \cup \tilde{N}_2)$ where $\tilde{N}_2 = \tilde{N}_L^{\max} \cup \tilde{N}_R^{\max} \cup \tilde{N}_L^{\text{ind}} \cup \tilde{N}_R^{\text{ind}} \cup \tilde{N}_*$. See Figure 4.6 for an illustration where the non-edges $\tilde{N}_*$ have been removed for clarity.

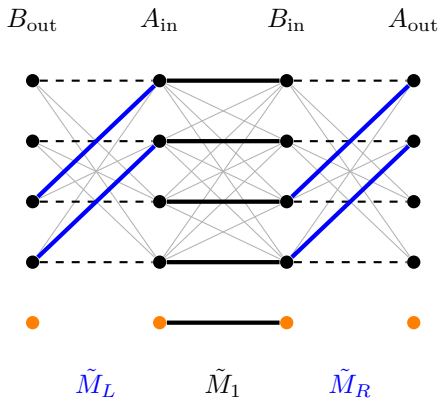


Figure 4.6: An illustration of the structure graph $\tilde{H}_2$. The thick blue edges represent the matching $\tilde{M}_2$ and the grey ones are the non-edges $\tilde{N}_2 \setminus \tilde{N}_*$. The orange vertices and their incident edge are isolated and only present if $|\tilde{M}_1|$ is odd. The black dashed edges represent a large maximum matching in a worst-case underlying graph.

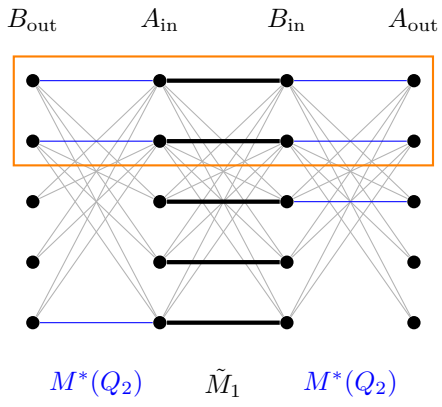


Figure 4.7: An example of the partitioning based on $M^*(Q_2)$. The blue edges represent the matching $M^*(Q_2)$. The vertices in the orange box represent part (A^+, B^+) and the remaining unboxed ones represent part (A^-, B^-). The grey edges show the additional non-edges asserted by the disjoint union.

Let Q_2 be an arbitrary query of round 2 and let $M^*(Q_2)$ be a maximum matching of

[4]Committing to the isolated edge and vertices simplifies the arguments in the subsequent round without affecting the approximation factor of the player's structure graphs.

(A, B, Q_2). By Assumption 4.21, we have that Q_2 only has edges in $A_{\text{in}} \times B_{\text{out}}$ and $A_{\text{out}} \times B_{\text{in}}$, which implies that the edges of $M^*(Q_2)$ either form vertex-disjoint length-3 or length-2 alternating paths w.r.t. $\tilde{M}_1$. As such, we partition the vertices of A and B to consider the length-3 and length-2 paths separately: Part (A^+, B^+) consists of all vertices that lie on a length-3 path, whereas part (A^-, B^-) consists of the remaining vertices, whose induced subgraph includes all the length-2 paths. See Figure 4.7 for an example of the partitioning.

Lemma 4.24. *$\tilde{H}_1$ is partitioned w.r.t. parts (A^+, B^+) and (A^-, B^-) into structure graphs $\tilde{H}_1^+ = \tilde{H}_1(A^+, B^+)$ and $\tilde{H}_1^- = \tilde{H}_1(A^-, B^-)$, respectively, where $\tilde{H}_1^+ \dot\cup \tilde{H}_1^-$ dominates $\tilde{H}_1$.*

Proof. Consider part (A^+, B^+) and its vertex-induced subgraph. Since (A^+, B^+) consists of the vertices of length-3 alternating paths w.r.t. $\tilde{M}_1$, the inclusion of the vertices of each path includes a matching edge from $\tilde{M}_1$ and two unmatched vertices, one in A_{out} and one in B_{out}. Thus, the matching edges $\tilde{M}_1^+ = \tilde{M}_1 \cap (A^+ \times B^+)$ and the non-edges $\tilde{N}_1^+ = \tilde{N}_1 \cap (A^+ \times B^+)$ are exactly the edges and non-edges of $\tilde{H}_1(A^+, B^+)$. Part (A^-, B^-) follows similarly since the remaining vertex-induced subgraph must have two unmatched vertices for every matching edge. Finally, since no edges of $\tilde{E}_1$ cross the partition, the disjoint union of each part dominates the original structure graph. □

With Lemma 4.24, the player's information in each part at the start of round 2 is exactly the respective generalisations of $\tilde{H}_1$. As such, we show that the respective generalisations of $\tilde{H}_2$ dominate the player's structure graphs H_2^+ and H_2^- learned in each part after query Q_2.

Lemma 4.25. *Any structure graph H_2^+ learned by the player is dominated by $\tilde{H}_2^+ = \tilde{H}_2(A^+, B^+)$.*

Proof. Let $Q_2^+ := Q_2 \cap (A^+ \times B^+)$ be the query edges relevant to part (A^+, B^+). By construction of the partition, $M^*(Q_2) \cap (A^+ \times B^+) \subseteq Q_2^+$ is a perfect matching in (A^+, B^+) such that every edge in $\tilde{M}_1^+$ is incident to two edges in $M^*(Q_2)$. Let W_L^+ be the perfect matching edges incident to $A_{\text{in}}^+ = A(\tilde{M}_1^+)$ and let W_R^+ be the ones incident to $B_{\text{in}}^+ = B(\tilde{M}_1^+)$. As such, $\tilde{M}_1^+ \cup W_L^+ \cup W_R^+$ is exactly the edges of the the $|\tilde{M}_1^+|$ many vertex-disjoint length-3 paths used to construct part (A^+, B^+). Therefore, we can construct matchings $M_L^+ \subseteq W_L^+$ and $M_R^+ \subseteq W_R^+$ of size $\left\lfloor \frac{|\tilde{M}_1^+|}{2} \right\rfloor$ such that $\tilde{M}_1^+ \cup M_L^+ \cup M_R^+$ has *no* length-3 paths. We then assert the maximality of $M_2^+ = M_L^+ \cup M_R^+$ by committing $N_L^+ = A_{\text{in}}^+ \backslash A(M_L^+) \times B_{\text{out}}^+ \backslash B(M_L^+)$ and $N_R^+ = A_{\text{out}}^+ \backslash A(M_R^+) \times B_{\text{in}}^+ \backslash B(M_R^+)$ as non-edges.

Finally, we let σ be a graph isomorphism[5] that relates this to $\tilde{H}_2^+$ where $M_L^+ = \sigma(\tilde{M}_L^+)$ and $M_R^+ = \sigma(\tilde{M}_R^+)$. Then, we have that $M_2^+ \subseteq \sigma(\tilde{M}_2^+)$ and $N_2^+ = N_L^+ \cup N_R^+ \subseteq \sigma(\tilde{N}_2^+)$. □

[5]Note that the player knows $\tilde{H}_1$ at the start of round 2 and $\tilde{H}_2$ is specified w.r.t. $\tilde{H}_1$; thus, the graph isomorphism from round 1 is implicitly considered in H_2 and $\tilde{H}_2$.

Lemma 4.26. *Any structure graph H_2^- learned by the player is dominated by $\tilde{H}_2^- = \tilde{H}_2(A^-, B^-)$.*

Proof. Let $Q_2^- := Q_2 \cap (A^- \times B^-)$ be the query edges relevant to part (A^-, B^-). By construction of the partition, $M^*(Q_2) \cap (A^- \times B^-) \subseteq Q_2^-$ is a maximum matching in (A^-, B^-) such that every edge in $\tilde{M}_1^-$ is incident to at most one edge in $M^*(Q_2)$. Let W_L^- be the matching edges incident to $A_{\text{in}}^- = A(\tilde{M}_1^-)$ and let W_R^- be the ones incident to $B_{\text{in}}^- = B(\tilde{M}_1^-)$. If $|W_L^-| \geq \left\lfloor \frac{|\tilde{M}_1^-|}{2} \right\rfloor$, then we let $M_L^- \subseteq W_L^-$ be of size $\left\lfloor \frac{|\tilde{M}_1^-|}{2} \right\rfloor$ and assert its maximality among the query edges $Q_2^- \cap (A_{\text{in}}^- \times B_{\text{out}}^-)$ by letting $N_L^- = A_{\text{in}}^- \backslash A(M_L^-) \times B_{\text{out}}^- \backslash B(M_L^-)$. Otherwise, if $|W_L^-| < \left\lfloor \frac{|\tilde{M}_1^-|}{2} \right\rfloor$, then we let $M_L^- = W_L^-$ which is trivially maximal; thus, $N_L^- = \emptyset$. We similarly consider W_R^- to construct the maximal matching M_R^- with non-edges N_R^-. Thus, there are no length-3 paths in $\tilde{M}_1^- \cup M_L^- \cup M_R^-$.

Finally, we let σ be a graph isomorphism that relates this to $\tilde{H}_2^-$ where $M_L^- \subseteq \sigma(\tilde{M}_L^-)$ and $M_R^- \subseteq \sigma(\tilde{M}_R^-)$; thus, we have that $M_2^- = M_L^- \cup M_R^- \subseteq \sigma(\tilde{M}_2^-)$ and $N_2^- = N_L^- \cup N_R^- \subseteq \sigma(\tilde{N}_2^-)$. □

By Lemmas 4.25 and 4.26, the player's overall information at the end of round 2 is dominated by the structure graph $\tilde{H}_2^+ \dot{\cup} \tilde{H}_2^-$.

Lemma 4.27. *The approximation factor of the structure graph $\tilde{H}_2^+ \dot{\cup} \tilde{H}_2^-$ is at most $\frac{1}{2} + \frac{2}{n}$.*

Proof. We claim that the largest matching is of size $|\tilde{M}_1|$ and that there exists a consistent graph G such that $\mu(G) = 2 \cdot |\tilde{M}_1| - 1$. Then, the approximation factor of $\tilde{H}_2^+ \dot{\cup} \tilde{H}_2^-$ is at most $\frac{|\tilde{M}_1|}{2 \cdot |\tilde{M}_1| - 1} = \frac{1}{2} + \frac{1}{4 \cdot |\tilde{M}_1| - 2} \leq \frac{1}{2} + \frac{1}{2 \cdot |\tilde{M}_1|} = \frac{1}{2} + \frac{2}{n}$ for large enough n.

We now prove the first claim. Since there are no augmenting paths in $\tilde{M}_1^+ \cup \tilde{M}_2^+$ or $\tilde{M}_1^- \cup \tilde{M}_2^-$, the largest output matching is $\tilde{M}_2^{\text{out}} = \tilde{M}_1^+ \cup \tilde{M}_1^- = \tilde{M}_1$ by Observation 4.19. It remains to prove the second claim. Since $|\tilde{M}_1| = \frac{n}{4}$ is odd, w.l.o.g., $|\tilde{M}_1^+|$ is odd and $|\tilde{M}_1^-|$ is even; hence, we have that $|\tilde{M}_2^+| = |\tilde{M}_1^+| - 1$ and $|\tilde{M}_2^-| = |\tilde{M}_1^-|$. Since both matchings in both parts are maximal and induced, the only unknown edges are those in $(A_{\text{in}} \times B_{\text{out}}) \cup (A_{\text{out}} \times B_{\text{in}})$ that have only one endpoint matched by $\tilde{M}_2^+$ or $\tilde{M}_2^-$ and are not incident to the isolated edge or vertices. Thus, we may use these to construct consistent graphs in $\tilde{H}_2^+$ and $\tilde{H}_2^-$ with maximum matchings of size $2 \cdot |\tilde{M}_2^+| + 1 = 2 \cdot |\tilde{M}_1^+| - 1$, which includes the isolated edge, and $2 \cdot |\tilde{M}_2^-| = 2 \cdot |\tilde{M}_1^-|$, respectively. By Observation 4.20, this gives the graph G as required. □

Overall, we have that any arbitrary query Q_2 can be used to construct the relevant partition of the vertices where Lemmas 4.25 to 4.27 always hold, thus proving Theorem 4.15-2.

4.3.3 Third Round

We continue to consider the partition w.r.t. the query Q_2 where the player now knows generalisations of $\tilde{H}_2$ in each part. As such, it is sufficient to find a structure graph $\tilde{H}_3$ that dominates $\tilde{H}_2$ and then apply the disjoint union as before. Note that we consider only the even case of $\tilde{H}_2$ since, by Assumption 4.21, the endpoints of the isolated edge e^*_{in} and the isolated vertices a_{out} and b_{out} in the odd case of $\tilde{H}_2$ (see Figure 4.6) are not endpoints of any edge in a third round query, thus reducing it to an even case of $\tilde{H}_2$.

With knowledge of an even case of $\tilde{H}_2$ at the start of round 3, the player is aware of two edge-disjoint maximal and induced matchings $\tilde{M}_1$ and $\tilde{M}_2$, both of which are half the size of a perfect matching, such that $\tilde{M}_1 \cup \tilde{M}_2$ is exactly the edges of $|\tilde{M}_1|$ many vertex-disjoint length-2 paths P, half of which have their endpoints in A while the other half have theirs in B. This implies that $\overline{A_{\text{out}}(P)}$ and $\overline{B_{\text{out}}(P)}$ each have $\frac{|\tilde{M}_1|}{2}$ many vertices where, for any set of vertices $U \subseteq A \cup B$, $U(P)$ denotes the U-endpoints of paths in P and $\overline{U(P)} := U \backslash U(P)$. By Assumption 4.21, any third round query may thus only contain the edges that either (a) *extend* a path in P, denoted by K^{ext}, or (b) provide a *replacement* edge for a path in P, denoted by K^{rep} where

$$K^{\text{ext}} := K^{\text{ext}}_L \cup K^{\text{ext}}_R = (A_{\text{in}}(P) \times B_{\text{out}}(P)) \cup (A_{\text{out}}(P) \times B_{\text{in}}(P)) \text{ and}$$
$$K^{\text{rep}} := K^{\text{rep}}_L \cup K^{\text{rep}}_R = \left(\overline{A_{\text{in}}(P)} \times \overline{B_{\text{out}}(P)}\right) \cup \left(\overline{A_{\text{out}}(P)} \times \overline{B_{\text{in}}(P)}\right).$$

We illustrate this in Figure 4.8.

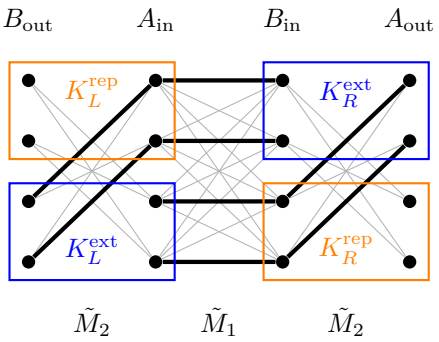


Figure 4.8: An illustration of the possible query edges by a player who knows $\tilde{H}_2$. The black edges represent the induced maximal matchings $\tilde{M}_1$ and $\tilde{M}_2$ and the grey ones are their corresponding non-edges $\tilde{N}_1$ and $\tilde{N}_2$. The possible query edges K^{ext}_L, K^{ext}_R, K^{rep}_L and K^{rep}_R are complete graphs on the vertices of their respective boxes.

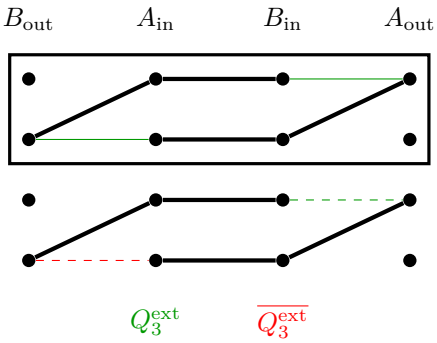


Figure 4.9: An example of the partitioning based on Q^{ext}_3. The green edges represent the edges in Q^{ext}_3 and the red edges represent the ones not in Q^{ext}_3, i.e., the edges in $\overline{Q^{\text{ext}}_3} = K^{\text{ext}} \backslash Q^{\text{ext}}_3$. The vertices in the box represent part (A°, B°) and the remaining unboxed ones represent part $(A^{\overline{\circ}}, B^{\overline{\circ}})$.

In general, the structure graph $\tilde{H}_2$ may be broken down into $\left\lfloor \frac{|\tilde{M}_1|}{4} \right\rfloor$ many identical vertex-disjoint gadgets $\mathcal{G}$ where each one consists of two length-2 paths with its endpoints in A, two length-2 paths with its endpoints in B, two vertices from $\overline{A_{\text{out}}(P)}$, and two vertices from $\overline{B_{\text{out}}(P)}$. Note that a single gadget is then exactly the generalisation of $\tilde{H}_2$ w.r.t. eight A-vertices and eight B-vertices; hence, Figure 4.8 is also an illustration of a single gadget where the only possible intra-gadget query edges are four sets of vertex-disjoint complete bipartite graphs on two A-vertices and two B-vertices. As such, we will require the following lemma:

Lemma 4.28. *Let $H_0 = (X, Y, \emptyset, \emptyset)$ be an empty structure graph where $|X| = |Y| = 2$ and let $g \in (X \times Y)$. Then, after any query $Q \subseteq (X \times Y)$, the player learns a structure graph that is dominated by $\tilde{H} = (X, Y, \{e\}, \{f\})$ such that $e \neq g$ and f are vertex-disjoint.*

Proof. If Q is the empty query, then no information is learned by the player and the claim holds. Otherwise, let $e \neq g \in Q$ be an arbitrary edge in the query, if one exists. Since there are only two A-vertices and two B-vertices, there exists only one possible edge f which is vertex-disjoint from e. If $f \in Q$ then we commit f as a non-edge, which makes the edge e a maximal matching. Otherwise, $f \notin Q$ and we have that e is already maximal. In the case where $e = g \in Q$ is the only query edge, we commit $f = g$ as a non-edge, which implies that the empty matching is maximal. □

At this stage, considering each gadget independently and dominating it by a structure graph would not imply a better than $(\frac{2}{3} + \Theta(\frac{1}{n}))$-approximation lower bound due to a hard query, which we discuss further in Section 4.3.4 for completeness. As such, we first partition the vertices w.r.t. to the query, obtaining desireable properties in each part of the partition.

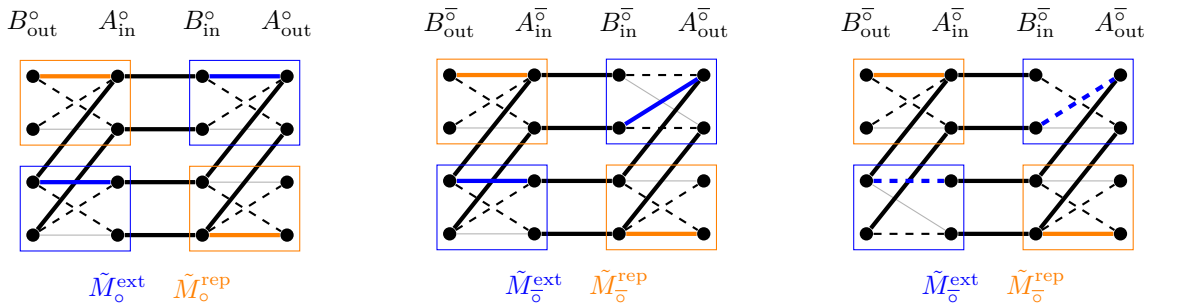
Let Q_3 be an arbitrary query of round 3. We partition the vertices A and B to consider the paths in P that maximally form vertex-disjoint 6-cycles with edges in $Q_3^{\text{ext}} = Q_3 \cap K^{\text{ext}}$ separately from the ones that do not: Part (A°, B°) consists of all vertices that lie on the vertex-disjoint 6-cycles, including, for each 6-cycle, a vertex from $\overline{A_{\text{out}}(P)}$ and one from $\overline{B_{\text{out}}(P)}$, whereas part $(A^{\overline{\circ}}, B^{\overline{\circ}})$ consists of the remaining vertices, whose induced subgraph includes all the paths in P that do not form any 6-cycles with each other using the query edges Q_3^{ext}. See Figure 4.9 for an example of the partitioning.

Lemma 4.29. *$\tilde{H}_2$ is partitioned w.r.t. parts (A°, B°) and $(A^{\overline{\circ}}, B^{\overline{\circ}})$ into structure graphs $\tilde{H}_2^\circ = \tilde{H}_2(A^\circ, B^\circ)$ and $\tilde{H}_2^{\overline{\circ}} = \tilde{H}_2(A^{\overline{\circ}}, B^{\overline{\circ}})$, respectively, where $\tilde{H}_2^\circ \dot{\cup} \tilde{H}_2^{\overline{\circ}}$ dominates $\tilde{H}_2$.*

Proof. Consider part (A°, B°) and its vertex-induced subgraph. Since (A°, B°) consists of the vertices of length-6 cycles each with a corresponding vertex unmatched by both matchings $\tilde{M}_1$ and $\tilde{M}_2$, we have that, for every two edges of $\tilde{M}_1$ included, two edges of $\tilde{M}_2$ – one incident to A_{in} and one to B_{in}– and two unmatched vertices – one in A_{out} and one in B_{out}– are added to (A°, B°). Therefore, it is a generalisation of an even case of $\tilde{H}_2$. Part $(A^{\overline{\circ}}, B^{\overline{\circ}})$ follows similarly

since the remaining vertices must have the same properties, which is also an even case since $\tilde{M}_1$ and $\tilde{M}_1^\circ$ are both even. Finally, since no edges of $\tilde{M}_1$ or $\tilde{M}_2$ cross the partition, $\tilde{H}_2^\circ \dot\cup \tilde{H}_2^{\overline{\circ}}$ dominates $\tilde{H}_2$. □

With Lemma 4.29, the player's information in each part at the start of round 3 is exactly the respective generalisations of $\tilde{H}_2$. As such, we can carefully construct the $\left\lfloor \frac{|\tilde{M}_1^\circ|}{4} \right\rfloor$ many gadgets $\mathcal{G}^\circ$ w.r.t. $\tilde{H}_2^\circ$ such that each gadget includes two vertex-disjoint 6-cycles using the intra-gadget query edges of $Q_3^{\text{ext}} \cap (A^\circ \times B^\circ)$. We can also arbitrarily construct the $\left\lfloor \frac{|\tilde{M}_1^{\overline{\circ}}|}{4} \right\rfloor$ many gadgets $\mathcal{G}^{\overline{\circ}}$ w.r.t. $\tilde{H}_2^{\overline{\circ}}$ where each gadget necessarily does not include any 6-cycles using the intra-gadget query edges of $Q_3^{\text{ext}} \cap (A^{\overline{\circ}} \times B^{\overline{\circ}})$. Next, we show that the structure graphs H_3° and $H_3^{\overline{\circ}}$ learned by the player in each part are dominated by distinct structure graphs $\tilde{H}_3^\circ$ and $\tilde{H}_3^{\overline{\circ}}$, respectively, after query Q_3.



(a) A gadget of the structure graph $\tilde{H}_3^\circ$.

(b) First case of a gadget of the structure graph $\tilde{H}_3^{\overline{\circ}}$.

(c) Second case of a gadget of the structure graph $\tilde{H}_3^{\overline{\circ}}$.

Figure 4.10: Illustrations of the gadgets of the structure graphs $\tilde{H}_3^\circ$ and $\tilde{H}_3^{\overline{\circ}}$, respectively. The thick blue edges (solid and dashed) represent the matchings $\tilde{M}_\circ^{\text{ext}}$ and $\tilde{M}_{\overline{\circ}}^{\text{ext}}$. The thick orange edges represent the matchings $\tilde{M}_\circ^{\text{rep}}$ and $\tilde{M}_{\overline{\circ}}^{\text{rep}}$. The grey edges represent the non-edges $\tilde{N}_\circ^{\max}$ and $\tilde{N}_{\overline{\circ}}^{\max}$. The dashed edges (black and blue) represent maximum matchings in worst-case underlying graphs.

Let $\tilde{M}_\circ^{\text{ext}} \subseteq K_\circ^{\text{ext}}$ be a matching such that each gadget in $\mathcal{G}^\circ$ has two edges that from a 6-cycle with two of the paths. Let $\tilde{M}_\circ^{\text{rep}} \subseteq K_\circ^{\text{rep}}$ be another matching such that each gadget has two edges, one from K_L^{rep} and one from K_R^{rep}, where at most one of them is incident to the 6-cycle. Let the edges of $K_\circ^{\text{ext}} \cup K_\circ^{\text{rep}}$ that are not within a gadget be the non-edges $\tilde{N}_\circ^{\text{gad}}$. We assert the maximality of the matching $\tilde{M}_3^\circ = \tilde{M}_\circ^{\text{ext}} \cup \tilde{M}_\circ^{\text{rep}}$ by committing the edges of $K_\circ^{\text{ext}} \cup K_\circ^{\text{rep}}$ within each gadget that have both endpoints unmatched by $\tilde{M}_3^\circ$ to be the non-edges $\tilde{N}_\circ^{\max}$. Then, we define $\tilde{H}_3^\circ = (A^\circ, B^\circ, \tilde{E}_2^\circ \cup \tilde{M}_3^\circ, \tilde{F}_2^\circ \cup \tilde{N}_3^\circ)$ where $\tilde{N}_3^\circ = \tilde{N}_\circ^{\text{gad}} \cup \tilde{N}_\circ^{\max}$. See Figure 4.10a for an illustration of a single gadget.

Lemma 4.30. *Any structure graph H_3° learned by the player is dominated by $\tilde{H}_3^\circ$.*

Proof. Let $Q_3^\circ = Q_3 \cap (A^\circ \times B^\circ)$ be any query w.r.t. part (A°, B°). At this point, each gadget of $\mathcal{G}^\circ$ is a generalisation of $\tilde{H}_2$ and receives a subset of the query edges Q_3° that are vertex-disjoint from the other gadgets. As such, it is sufficient to consider any arbitrary query $Q'_\circ$ w.r.t. a single gadget. In general, $Q'_\circ$ can be partitioned into four smaller parts: $Q_L^{\text{ext}} = Q'_\circ \cap K_L^{\text{ext}}$, $Q_R^{\text{ext}} = Q'_\circ \cap K_R^{\text{ext}}$, $Q_L^{\text{rep}} = Q'_\circ \cap K_L^{\text{rep}}$ and $Q_R^{\text{rep}} = Q'_\circ \cap K_R^{\text{rep}}$, each being any arbitrary query w.r.t. two A-vertices and two B-vertices.

By Lemma 4.28, we have that the player learns at most the edges $e_L^{\text{rep}} \in K_L^{\text{rep}}$ and $e_R^{\text{rep}} \in K_R^{\text{rep}}$, possibly with the non-edges $f_L^{\text{rep}} \in K_L^{\text{rep}}$ and $f_R^{\text{rep}} \in K_R^{\text{rep}}$, after the queries Q_L^{rep} and Q_R^{rep}, respectively. Therefore, with a slight abuse of notation, the player learns the matching $M_\circ^{\text{rep}} \subseteq \tilde{M}_\circ^{\text{rep}} = \{e_L^{\text{rep}}, e_R^{\text{rep}}\}$ and the non-edges $N_\circ^{\text{rep}} \subseteq \tilde{N}_\circ^{\text{rep}} = \{f_L^{\text{rep}}, f_R^{\text{rep}}\}$. Recall that, by construction of the partition, the gadget has two vertex-disjoint 6-cycles c_e and c_f that require the edges $e_L^{\text{ext}}, f_L^{\text{ext}} \in Q_L^{\text{ext}}$ and $e_R^{\text{ext}}, f_R^{\text{ext}} \in Q_R^{\text{ext}}$, respectively. W.l.o.g. let c_e be the cycle that is not incident to both e_L^{rep} and e_R^{rep}. Then, the player learns the matching $M_\circ^{\text{ext}} = \tilde{M}_\circ^{\text{ext}} = \{e_L^{\text{ext}}, e_R^{\text{ext}}\}$ and the non-edges $N_\circ^{\text{ext}} = \tilde{N}_\circ^{\text{ext}} = \{f_L^{\text{ext}}, f_R^{\text{ext}}\}$.

Finally, to assert the maximality of the matchings overall, let $N_\circ^{\text{gad}}$ be the query edges Q_3° that are not within a gadget of $\mathcal{G}^\circ$; hence, $N_\circ^{\text{gad}} \subseteq \tilde{N}_\circ^{\text{gad}}$. It follows that any structure graph H_3° learned w.r.t. the player's query Q_3° is dominated by $\tilde{H}_3$, without any further graph isomorphisms. □

Let $\tilde{M}_{\overline{\circ}}^{\text{ext}} \subseteq K_{\overline{\circ}}^{\text{ext}}$ be a matching such that each gadget in $\mathcal{G}^{\overline{\circ}}$ has two edges that form a length-8 path with three of the paths, leaving the other path unmatched. Let $\tilde{M}_{\overline{\circ}}^{\text{rep}} \subseteq K_{\overline{\circ}}^{\text{rep}}$ be another matching such that each gadget has two edges, one from K_L^{rep} and one from K_R^{rep}, where one of them is incident to the path unmatched by $\tilde{M}_{\overline{\circ}}^{\text{ext}}$. Let the edges of $K_{\overline{\circ}}^{\text{ext}} \cup K_{\overline{\circ}}^{\text{rep}}$ that are not within a gadget be the non-edges $\tilde{N}_{\overline{\circ}}^{\text{gad}}$. We assert the maximality of the matching $\tilde{M}_{\overline{\circ}}^{\text{ext}}$ among the edges of $K_{\overline{\circ}}^{\text{ext}}$ with the non-edges $\tilde{N}_{\overline{\circ}}^{\text{ext}}$ in two distinct cases of a gadget: $\tilde{N}_{\overline{\circ}}^{\text{ext}}$ is such that each gadget either has (1) the only two edges of $K_{\overline{\circ}}^{\text{ext}}$ with both endpoints unmatched by $\tilde{M}_{\overline{\circ}}^{\text{ext}}$, or (2) two edges of $K_{\overline{\circ}}^{\text{ext}}$ that form a length-8 path and are both incident to only the A-vertices or only the B-vertices of $\tilde{M}_{\overline{\circ}}^{\text{ext}}$. Note that the gadgets are indeed maximal since, by construction of part $(A^{\overline{\circ}}, B^{\overline{\circ}})$, there are no 6-cycles among its query edges. We assert the maximality of the matching $\tilde{M}_{\overline{\circ}}^{\text{rep}}$ among the edges of $K_{\overline{\circ}}^{\text{rep}}$ in each gadget by committing its edges that have both endpoints unmatched by $\tilde{M}_{\overline{\circ}}^{\text{rep}}$ to be the non-edges $\tilde{N}_{\overline{\circ}}^{\text{rep}}$. Then, we define $\tilde{H}_3^{\overline{\circ}} = (A^{\overline{\circ}}, B^{\overline{\circ}}, \tilde{E}_2^{\overline{\circ}} \cup \tilde{M}_3^{\overline{\circ}}, \tilde{F}_2^{\overline{\circ}} \cup \tilde{N}_3^{\overline{\circ}})$ where $\tilde{N}_3^{\overline{\circ}} = \tilde{N}_{\overline{\circ}}^{\text{gad}} \cup \tilde{N}_{\overline{\circ}}^{\max}$. See Figures 4.10b and 4.10c for illustrations of the two cases of a gadget.

Lemma 4.31. *Any structure graph $H_3^{\overline{\circ}}$ learned by the player is dominated by $\tilde{H}_3^{\overline{\circ}}$.*

Proof. Let $Q_3^{\overline{\circ}} = Q_3 \cap (A^{\overline{\circ}} \times B^{\overline{\circ}})$ be any query w.r.t. part $(A^{\overline{\circ}}, B^{\overline{\circ}})$. At this point, each gadget of $\mathcal{G}^{\overline{\circ}}$ is a generalisation of $\tilde{H}_2$ and receives a subset of the query edges $Q_3^{\overline{\circ}}$ that are vertex-disjoint from the other gadgets. As such, it is sufficient to consider any arbitrary query $Q'_{\overline{\circ}}$ w.r.t. a

single gadget. In general, $Q'_{\overline{\circ}}$ can be partitioned into four smaller parts: $Q_L^{\text{ext}} = Q'_{\overline{\circ}} \cap K_L^{\text{ext}}$, $Q_R^{\text{ext}} = Q'_{\overline{\circ}} \cap K_R^{\text{ext}}$, $Q_L^{\text{rep}} = Q'_{\overline{\circ}} \cap K_L^{\text{rep}}$ and $Q_R^{\text{rep}} = Q'_{\overline{\circ}} \cap K_R^{\text{rep}}$, each being any arbitrary query w.r.t. two A-vertices and two B-vertices.

By Lemma 4.28, we have that the player learns at most the edges $e_L^{\text{rep}} \in K_L^{\text{rep}}$ and $e_R^{\text{rep}} \in K_R^{\text{rep}}$, possibly with the non-edges $f_L^{\text{rep}} \in K_L^{\text{rep}}$ and $f_R^{\text{rep}} \in K_R^{\text{rep}}$, after the queries Q_L^{rep} and Q_R^{rep}, respectively. Therefore, with a slight abuse of notation, the player learns the matching $M_{\overline{\circ}}^{\text{rep}} \subseteq \tilde{M}_{\overline{\circ}}^{\text{rep}} = \{e_L^{\text{rep}}, e_R^{\text{rep}}\}$ and the non-edges $N_{\overline{\circ}}^{\text{rep}} \subseteq \tilde{N}_{\overline{\circ}}^{\text{rep}} = \{f_L^{\text{rep}}, f_R^{\text{rep}}\}$. Then, edge e_L^{rep} is incident to a path p with its endpoints in $B^{\overline{\circ}}$ while edge e_R^{rep} is incident to a path q with its endpoints in $A^{\overline{\circ}}$. Furthermore, we have that there exists a unique edge $g_L^{\text{ext}} \in K_L^{\text{ext}}$ and a unique edge $g_R^{\text{ext}} \in K_R^{\text{ext}}$ each of which are incident to both p and q – we never want these edges to belong to the player's maximal matching $M_{\overline{\circ}}^{\text{ext}}$.

We now consider the pair of queries Q_L^{ext} and Q_R^{ext} that, by construction of part $(A^{\overline{\circ}}, B^{\overline{\circ}})$, do not contain edges that form 6-cycles with any paths in $P^{\overline{\circ}}$. This implies that if $g_L^{\text{ext}} \in Q_L^{\text{ext}}$ (or $g_R^{\text{ext}} \in Q_R^{\text{ext}}$) then $g_R^{\text{ext}} \notin Q_R^{\text{ext}}$ (resp. $g_L^{\text{ext}} \notin Q_L^{\text{ext}}$). Furthermore, since every pair of queries Q_L^{ext} and Q_R^{ext} has a symmetrical copy, we only need to consider the pairs of queries where $g_L^{\text{ext}} \notin Q_L^{\text{ext}}$.

If either query is empty, by Lemma 4.28, we have that the edges, which avoid g_R^{ext}, and non-edges learned are thus subsets of a first case gadget in $\tilde{H}_3^{\overline{\circ}}$. Otherwise, both queries are non-empty and there exists query edges $e_L \neq g_L^{\text{ext}} \in Q_L^{\text{ext}}$ and $e_R \in Q_R^{\text{ext}}$. We then have that they either form a length-8 path in the gadget or do not. Consider first the simpler latter case where the query edges necessarily form precisely two length-5 paths. Notice that there can be no other query edges since any other query edge would form a length-8 path or a 6-cycle. Let $M_{\overline{\circ}}^{\text{ext}} = \{e_L^{\text{ext}}\}$ and $N_{\overline{\circ}}^{\text{ext}} = \{f_R^{\text{ext}}\}$ where $e_L^{\text{ext}} = e_L$ and $f_R^{\text{ext}} = e_R$. $M_{\overline{\circ}}^{\text{ext}}$ is naturally maximal, and the edges and non-edges learned are a subset of the second case gadget in $\tilde{H}_3^{\overline{\circ}}$.

Consider now the former case where the query edges $e_L \neq g_L^{\text{ext}}$ and e_R form a length-8 path. If $e_R \neq g_R^{\text{ext}}$, let $M_{\overline{\circ}}^{\text{ext}} = \{e_L^{\text{ext}}, e_R^{\text{ext}}\}$ where $e_L^{\text{ext}} = e_L$ and $e_R^{\text{ext}} = e_R$. Since $e_L^{\text{ext}} \neq g_L^{\text{ext}}$ the endpoints of either p or q in the gadget remain unmatched by $M_{\overline{\circ}}^{\text{ext}}$. Then, if they exist, we add the edges $f_L^{\text{ext}} \in Q_L^{\text{ext}}$ and $f_R^{\text{ext}} \in Q_R^{\text{ext}}$ that are unmatched by $M_{\overline{\circ}}^{\text{ext}}$ to an initially empty set of non-edges $N_{\overline{\circ}}^{\text{ext}}$; hence, $M_{\overline{\circ}}^{\text{ext}}$ is maximal and the edges and non-edges learned are a subset of the first case gadget in $\tilde{H}_3^{\overline{\circ}}$. Otherwise, we have that $e_R = g_R^{\text{ext}}$ and let $N_{\overline{\circ}}^{\text{ext}} = \{f_L^{\text{ext}}, f_R^{\text{ext}}\}$ where $f_L^{\text{ext}} = e_L$ and $f_R^{\text{ext}} = e_R$. Any possible remaining query edges must be either only incident to or only not incident to $N_{\overline{\circ}}^{\text{ext}}$ since they may not form any 6-cycles. We pick at most one from Q_L^{ext} and one from Q_R^{ext} then add them to an initially empty matching $M_{\overline{\circ}}^{\text{ext}}$, which is thus maximal and, if both exist, forms a length-8 path that leaves the endpoints of either p or q in the gadget unmatched since $M_{\overline{\circ}}^{\text{ext}}$ does not contain g_L^{ext} or g_R^{ext}. If $M_{\overline{\circ}}^{\text{ext}}$ and $N_{\overline{\circ}}^{\text{ext}}$ are incident to each other, then the edges and non-edges learned are a subset of the second case gadget in $\tilde{H}_3^{\overline{\circ}}$. Otherwise, they are a subset of the first case gadget in $\tilde{H}_3^{\overline{\circ}}$. □

By Lemmas 4.30 and 4.31, the player, whose structure graph is $\tilde{H}_2$ at the start of round 2, has its structure graph dominated by $\tilde{H}_3^\circ \dot\cup \tilde{H}_3^{\overline{\circ}}$ by the end of round 3. Note that, since there are only intra-gadget edges, $\tilde{H}_3^\circ \dot\cup \tilde{H}_3^{\overline{\circ}}$ is made up of the collection of gadgets $\mathcal{G} = \mathcal{G}^\circ \cup \mathcal{G}^{\overline{\circ}}$ where all edges that are not within a gadget are non-edges. As such, each gadget is either a gadget from $\tilde{H}_3^\circ$, the first case gadget from $\tilde{H}_3^{\overline{\circ}}$, or the second case gadget from $\tilde{H}_3^{\overline{\circ}}$. It follows then that, since $|\tilde{M}_1|$ is even, there are at least $\left\lfloor \frac{|\tilde{M}_1^\circ|}{4} \right\rfloor + \left\lfloor \frac{|\tilde{M}_1^{\overline{\circ}}|}{4} \right\rfloor \geq \frac{|\tilde{M}_1^\circ|}{4} - \frac{1}{2} + \frac{|\tilde{M}_1^{\overline{\circ}}|}{4} - \frac{1}{2} = \frac{|\tilde{M}_1|}{4} - 1$ many gadgets in $\mathcal{G}$ and, since each gadget has exactly four edges of $\tilde{M}_1 = \tilde{M}_1^\circ \cup \tilde{M}_1^{\overline{\circ}}$, there are at most $\frac{|\tilde{M}_1|}{4}$ many gadgets in $\mathcal{G}$.

Lemma 4.32. *The largest matching in $\tilde{H}_3^\circ \dot\cup \tilde{H}_3^{\overline{\circ}}$ is of size at most $\frac{5 \cdot |\tilde{M}_1|}{4}$.*

Proof. Since each gadget is vertex-disjoint, we only need to consider the number of edges that each case of a gadget contributes to a largest output matching $\tilde{M}_3^{\text{out}}$. By Berge's theorem or similar, every edge with an endpoint of degree 1 is included in a largest output matching and thus it is easy to see that each gadget contributes exactly 5 edges to $\tilde{M}_3^{\text{out}}$. Finally, any edge of $\tilde{M}_1$ not included in a gadget contributes less advantageously to a largest matching, hence; we assume that all of them form part of a gadget, which implies that $|\mathcal{G}| = \frac{|\tilde{M}_1|}{4}$ and the result. □

Lemma 4.33. *There exists a graph consistent with $\tilde{H}_3^\circ \dot\cup \tilde{H}_3^{\overline{\circ}}$ that has a maximum matching of size at least $2 \cdot |\tilde{M}_1| - 4$.*

Proof. Any graph consistent with $\tilde{H}_3^\circ \dot\cup \tilde{H}_3^{\overline{\circ}}$ may only consist of the edges within the vertex-disjoint gadgets that are not non-edges and the edges $\tilde{M}_1 \cup \tilde{M}_2$. As such, we first construct a maximum matching w.r.t. to each case of a gadget independently. Observe that all cases of a gadget, on 16 vertices, can be partitioned into 4 vertex-disjoint sets such that each consists of two A-vertices and two B-vertices, and they respectively correspond to the edges K_L^{ext}, K_R^{ext}, K_L^{rep} and K_R^{rep} unknown to the player who knows $\tilde{H}_2$. After query Q_3, at most a single non-edge f in each set is learned; hence, the remaining two edges incident to f can be used to construct a perfect matching in each set, which is thus a perfect matching of size 8 in each case of the gadget (see Figure 4.10). It remains to consider the maximum matching w.r.t. the edges not in the gadgets, which consists of $x = 0, 2, 4$ many length-2 paths from $\tilde{M}_1 \cup \tilde{M}_2$ since $|\tilde{M}_1|$ is even. In particular, we have that $|\mathcal{G}| = \frac{|\tilde{M}_1| - x}{4}$. Since a single edge from each of the x many paths belongs to any maximum matching, we may construct a consistent graph with a maximum matching of size $8 \cdot |\mathcal{G}| + x = 2 \cdot |\tilde{M}_1| - x \geq 2 \cdot |\tilde{M}_1| - 4$. □

Overall, we have that the structure graph learned by the player after round 1 is dominated by $\tilde{H}_1$, for any arbitrary query Q_1. Then, in round 2, any arbitrary query Q_2 made by the player partitions the vertices into parts (A^+, B^+) and (A^-, B^-) such that the structure graphs learned in each part is dominated by the respective generalisations of $\tilde{H}_2$, i.e., $\tilde{H}_2^+$ and $\tilde{H}_2^-$, by the end of round 2. In round 3, each part is dominated by the respective generalisations of

$\tilde{H}_3^{\circ} \dot{\cup} \tilde{H}_3^{\bar{\circ}}$, which we denote as $\tilde{H}_3^+$ and $\tilde{H}_3^-$, respectively, for any arbitrary query Q_3. As such, the player's overall information by the end of round 3 is $\tilde{H}_3^+ \dot{\cup} \tilde{H}_3^-$ for any arbitrary sequence of queries Q_1, Q_2, Q_3. Finally, we prove Lemma 4.34, which implies Theorem 4.15-3.

Lemma 4.34. *The approximation factor of the structure graph $\tilde{H}_3^+ \dot{\cup} \tilde{H}_3^-$ is at most $\frac{5}{8} + \frac{12}{n}$.*

Proof. Recall that $\frac{n}{4}$ is odd; hence, $\tilde{H}_2^+$ and $\tilde{H}_2^-$ are such that, w.l.o.g., $|\tilde{M}_1^+|$ is even and $|\tilde{M}_1^-|$ is odd, and are dominated by the respective generalisations of $\tilde{H}_3^{\circ} \dot{\cup} \tilde{H}_3^{\bar{\circ}}$, that is, $\tilde{H}_3^+$ and $\tilde{H}_3^-$. As such, we consider both cases, particularly paying attention to $\tilde{H}_3^-$ and considering its isolated edge and vertices. It follows by Lemma 4.32 that $\tilde{H}_3^+$ has a largest matching of size at most $\frac{5 \cdot |\tilde{M}_1^+|}{4}$; however, $\tilde{H}_3^-$ has one of size at most $\frac{5 \cdot |\tilde{M}_1^-|}{4} - \frac{1}{4}$. Then, Lemma 4.33 immediately implies that $\tilde{H}_3^+$ has a consistent graph G_3^+ such that $\mu(G_3^+) \geq 2 \cdot |\tilde{M}_1^+| - 4$; however, it implies that $\tilde{H}_3^-$ has a consistent graph G_3^- such that $\mu(G_3^-) \geq 2 \cdot |\tilde{M}_1^-| - 5$. Finally, by Observations 4.19 and 4.20, the approximation factor of $\tilde{H}_3^+ \dot{\cup} \tilde{H}_3^-$ is at most $\frac{\frac{5}{4}|\tilde{M}_1| - \frac{1}{4}}{2 \cdot |\tilde{M}_1| - 9} \leq \frac{5}{8} + \frac{3}{|\tilde{M}_1|} = \frac{5}{8} + \frac{12}{n}$ for large enough n. □

4.3.4 Motivation of the lower bound partitioning w.r.t. Q_3

In this subsection, we present a hard third round query Q_3' for a single gadget of $\mathcal{G}$ w.r.t. $\tilde{H}_2$ that motivates the partitioning of the vertices w.r.t. any arbitrary third round query Q_3 used in Section 4.3.3. In particular, we show that directly considering vertex-disjoint gadgets w.r.t. $\tilde{H}_2$, without first partitioning the vertices, can not lead to proving a better than $(\frac{2}{3} + \Theta(\frac{1}{n}))$-approximation lower bound.

Observe first that Q_3' can be considered as four smaller parts $Q_L^{\text{ext}} = Q_3' \cap K_L^{\text{ext}}$, $Q_R^{\text{ext}} = Q_3' \cap K_R^{\text{ext}}$, $Q_L^{\text{rep}} = Q_3' \cap K_L^{\text{rep}}$, and $Q_R^{\text{rep}} = Q_3' \cap K_R^{\text{rep}}$ where each part may be any arbitrary query on two A-vertices and two B-vertices. Let (e_A, f_A) be the edges of a length-2 path of the gadget with its endpoints in A and let (e_B, f_B) be the edges of one with its endpoints in B such that $e_A, e_B \in \tilde{M}_1$. Then, Q_L^{ext} includes both edges incident to e_A, Q_R^{ext} includes both edges incident to e_B, Q_L^{rep} includes both edges incident to e_B, and Q_R^{rep} includes both edges incident to e_A. See Figure 4.11 for an illustration.

It is easy to see that this hard query Q_3' will always produce a matching that is at least a $\frac{2}{3}$-approximation, which occurs when Q_L^{ext} and Q_R^{ext} are committed as non-edges implying that the largest output matching M_3^{out} in each gadget is of size 4 while its maximum matching M^* is of size 6. Then, since all except a constant number of vertices can be used to construct the gadgets in $\mathcal{G}$, a hard query Q_3 such that the intra-gadget query edges are exactly Q_3' for each gadget in $\mathcal{G}$ results in a structure graph H_3 with approximation factor $\frac{2}{3} + \Theta(\frac{1}{n})$.

The partitioning of the vertices that we consider in Section 4.3.3 immediately deals with such a hard query. The hard query described consists of gadgets that have two length-2 paths, (e_A, f_A) and (e_B, f_B), that form a length 6-cycle w.r.t. Q_3' whereas the other two length-2 paths

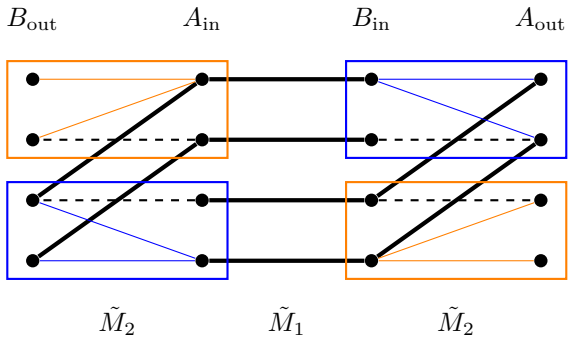


Figure 4.11: An illustration of a hard query Q'_3 for a single gadget of $\mathcal{G}$ w.r.t. $\tilde{H}_2$. The thick black edges represent the matching edges $\tilde{M}_1$ and $\tilde{M}_2$. The blue edges in each blue box represents the query edges Q_L^{ext} and Q_R^{ext}, respectively. The orange edges in each orange box represents the query edges Q_L^{rep} and Q_R^{rep}, respectively. The dashed black edges are the edges of $M^* \backslash \tilde{M}_1$.

necessarily do not form a 6-cycle. The partitioning into parts (A°, B°) and $(A^{\overline{\circ}}, B^{\overline{\circ}})$ would, however, consider pairs of length-2 paths that form 6-cycles w.r.t. Q'_3 separately from the ones that do not; thus, by carefully constructing the gadgets, this hard query can be avoided in all except at most one gadget.

4.4 Conclusion

In this chapter, we gave tight results on the approximation factor achievable by deterministic algorithms in the maximal matching edge-query model for up to 3 rounds. Our main result is a 0.625-approximation algorithm for MBM, which operates in three query rounds, and we proved that this is best possible. This algorithm can be implemented in the semi-streaming model and constitutes an improvement over the previously best 3-pass algorithm with approximation factor 0.6111 by Feldman and Szarf [FS22]. The best approximation factors achievable in one and two rounds are $\frac{1}{2}$ and $\frac{1}{2} + \Theta(\frac{1}{n})$, respectively, even in general graphs.

We conclude with three open questions:

1. *Randomization.* Our paper only considers deterministic query algorithms. Does randomization allow us to improve upon the results obtained in this chapter?
2. *Adaptivity.* The algorithms considered in this chapter are *adaptive* in the sense that the ith query can depend on the maximal matchings returned in rounds $1, \ldots, i-1$. Can we obtain interesting results if we allow multiple *non-adaptive* queries, i.e., queries that do not depend on the output produced by other queries?
3. *Semi-streaming Algorithms.* Is there a 3-pass semi-streaming algorithm for MBM with approximation factor better than 0.625 (that necessarily cannot be implemented as a deterministic edge-query algorithm)?

5 Greedy-first Approximate MBM

This chapter has been published as the following work:

- [KN21]: Christian Konrad and Kheeran K. Naidu. On Two-Pass Streaming Algorithms for Maximum Bipartite Matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021.*

Abstract

We study two-pass streaming algorithms for Maximum Bipartite Matching (MBM). Most known two-pass streaming algorithms for MBM operate in a similar fashion: They compute a maximal matching in the first pass and find 3-augmenting paths in the second in order to augment the matching found in the first pass. Our aim is to explore the limitations of this approach and to determine whether current techniques can be used to further improve the state-of-the-art algorithms. We give the following results:

We show that every two-pass streaming algorithm that solely computes a maximal matching in the first pass and outputs a $(\frac{2}{3}+\epsilon)$-approximation requires $n^{1+\Omega(\frac{1}{\log\log n})}$ space, for every $\epsilon > 0$, where n is the number of vertices of the input graph. This result is obtained by extending the Ruzsa-Szemerédi graph construction of [Goel et al., SODA'12] so as to ensure that the resulting graph has a close to perfect matching, the key property needed in our construction. This result may be of independent interest.

Furthermore, we combine the two main techniques, i.e., subsampling followed by the Greedy matching algorithm [Konrad, MFCS'18] which gives a $2-\sqrt{2} \approx 0.5857$-approximation, and the computation of *degree-bounded semi-matchings* [Esfandiari et al., ICDMW'16][Kale and Tirodkar, APPROX'17] which gives a $\frac{1}{2} + \frac{1}{12} \approx 0.5833$-approximation, and obtain a meta-algorithm that yields Konrad's and Esfandiari et al.'s algorithms as special cases. This unifies two strands of research. By optimizing parameters, we discover that Konrad's algorithm is optimal for the implied class of algorithms and, perhaps surprisingly, that there is a second optimal algorithm. We show that the analysis of our meta-algorithm is best possible. Our results imply that further improvements, if possible, require new techniques.

5.1 Introduction

In this chapter, we focus on MBM in the insertion-only semi-streaming model of computation. The currently best one-pass semi-streaming algorithm for MBM is the Greedy matching algorithm (depicted in Figure 2). Greedy processes the edges of a graph in arbitrary order and inserts the current edge into an initially empty matching if possible. It produces a maximal matching, which is known to be at least half the size of a maximum matching, and constitutes a $\frac{1}{2}$-approximation semi-streaming algorithm for MBM. It is a long-standing open question whether Greedy is optimal for the class of semi-streaming algorithms or whether an improved approximation ratio is possible. Progress has been made on the lower bound side ([GKK12, Kap13, Kap21]), ruling out semi-streaming algorithms with approximation ratio better than $\frac{1}{1+\ln 2} \approx 0.5906$ [Kap21].

Konrad et al. [KMM12] were the first to show that an approximation ratio better than $\frac{1}{2}$ can be achieved if two passes over the input are allowed, and further successive improvements [KT17, EHM16, Kon18b] led to a randomized two-pass semi-streaming algorithm with an approximation factor of $2 - \sqrt{2} \approx 0.58578$ [Kon18b] (see Table 5.1 for an overview of two-pass algorithms for MBM).

Approximation Factor	Reference	Comment
$\frac{1}{2} + 0.019 = 0.5190$	Konrad et al. [KMM12]	randomized
$\frac{1}{2} + \frac{1}{16} = 0.5625$	Kale and Tirodkar [KT17]	deterministic
$\frac{1}{2} + \frac{1}{12} \approx 0.5833$	Esfandiari et al. [EHM16]	deterministic
$2 - \sqrt{2} \approx 0.5857$	Konrad [Kon18b]	randomized

Table 5.1: Two-pass semi-streaming algorithms for Maximum Bipartite Matching.

Most known two-pass streaming algorithms proceed in a similar fashion. In the first pass, they run Greedy in order to compute a maximal matching M. In the second pass, they pursue different strategies to compute additional edges F that allow them to increase the size of M. Two techniques for computing the edge set F have been used:

1. **Subsampling and** Greedy [Kon18b] (see also [KMM12]): Given a bipartite graph $G = (A, B, E)$ and a first-pass maximal matching M, they first subsample the edges M with probability p and obtain a matching $M' \subseteq M$. Then, in the second pass, they compute Greedy matchings M_L and M_R on subgraphs $G_L = G[A(M') \cup \overline{B(M)}]$ and $G_R = G[\overline{A(M)} \cup B(M')]$, respectively, where $A(M')$ are the matched A-vertices in M', $\overline{B(M)}$ are the B vertices not matched in M, and $B(M')$ and $\overline{A(M)}$ are defined similarly. It can be seen that if M is relatively small, then $M' \cup M_L \cup M_R$ contains many disjoint

Algorithm 2 GREEDY Matching

Input: Graph $G = (A, B, E)$

1: $M \leftarrow \varnothing$
2: **for each** edge $e \in E$ (arbitrary order)
3: **if** $M \cup \{e\}$ is a matching
4: $M \leftarrow M \cup \{e\}$
5: **return** M

Algorithm 3 GREEDY$_d$ Semi-Matching

Input: Graph $G = (A, B, E)$, integer d

1: $S \leftarrow \varnothing$
2: **for each** edge $ab \in E$ (arbitrary order)
3: **if** $\deg_S(a) = 0$ **and** $\deg_S(b) < d$
4: $S \leftarrow S \cup \{ab\}$
5: **return** S

3-augmenting paths. Setting $p = \sqrt{2} - 1$ yields the approximation factor $2 - \sqrt{2}$.

2. **Semi-matchings and** GREEDY$_d$ [KT17, EHM16]: Given a bipartite graph $G = (A, B, E)$ and a first-pass maximal matching M, the second pass consists of finding *degree-d-constrained semi-matchings* S_L and S_R on subgraphs $G_L = G[A(M) \cup \overline{B(M)}]$ and $G_R = G[\overline{A(M)} \cup B(M)]$, respectively, using the algorithm GREEDY$_d$ (as depicted in Figure 3). A degree-d-constrained semi-matching in a bipartite graph is a subset of edges $S \subseteq E$ such that $\deg_S(a) \leq 1$ and $\deg_S(b) \leq d$, for every $a \in A$ and $b \in B$ or vice versa[1]. Similar to the method above, it can be seen that if the matching M is relatively small, $M \cup S_L \cup S_R$ contains many disjoint 3-augmenting paths. The setting $d = 3$ yields the approximation factor $\frac{1}{2} + \frac{1}{12}$.

Our Results. In this chapter, we explore the limitations of this approach and investigate whether current techniques can be used to further improve the state-of-the-art.

Our first result is a limitation result on the approximation factor achievable by algorithms that follow the scheme described above:

Theorem 5.1 (simplified, cf. Theorem 4)**.** *Every two-pass semi-streaming algorithm for MBM that solely runs* GREEDY *in the first pass has an approximation factor of at most* $\frac{2}{3}$.

Our result builds upon a result by Goel et al. [GKK12] who proved that the lower bound of Theorem 5.1 applies to one-pass streaming algorithms. Their construction relies on the existence of dense *Ruzsa-Szemerédi* graphs with large induced matchings, i.e., bipartite $2n$-vertex graphs $G = (A, B, E)$ with $|A| = |B| = n$ whose edge sets can be partitioned into disjoint induced matchings such that each matching is of size at least $(\frac{1}{2} - \delta) \cdot n$, for some small δ. Our construction requires similarly dense RS-graphs with equally large matchings; however, in addition to these properties, our RS-graphs must contain a *near-perfect* matching, i.e., a matching that matches all but a small constant fraction of the vertices. To this end, we augment the RS-graph construction by Goel et al.: We show that, for each induced matching

[1] The usual definition of a semi-matching requires $\deg_S(a) = 1$, for every $a \in A$ (e.g. [FLN14, KR16]). This property is not required here, but we stick to this term for ease of notation.

M in Goel et al.'s construction, we can add a matching M' to the construction without violating the induced matching property such that $M \cup M'$ forms a near-perfect matching. We believe this result may be of independent interest.

Next, we combine the subsampling and semi-matching techniques to give a meta-algorithm that yields Konrad's and Esfandiari et al.'s algorithms as special cases, thereby unifying two strands of research. Our meta-algorithm is parameterised by a sampling probability $0 < p \leq 1$ and an integral degree bound $d \geq 1$. First, similarly to the subsampling technique (Technique 1), the edges of the first-pass matching M are subsampled independently with probability p, which yields a subset $M' \subseteq M$. Next, similarly to the semi-matching technique (Technique 2), incomplete semi-matchings S_L and S_R with degree bounds d are computed; now, however, in the subgraphs $G'_L = G[A(M') \cup \overline{B(M)}]$ and $G'_R = G[\overline{A(M)} \cup B(M')]$. The algorithm then outputs the largest matching among the edges $M \cup S_L \cup S_R$.

As our second result, we establish the approximation factor of our meta-algorithm:

Theorem 5.2 (simplified, cf. Theorem 3)**.** *Combining the subsampling and semi-matching techniques yields a two-pass semi-streaming algorithm for MBM with approximation factor*

$$\begin{cases} \frac{1}{2} + (\frac{1}{d+p} - \frac{1}{2d}) \cdot p, & \text{if } p \leq d(\sqrt{2}-1) \\ \frac{1}{2} + \frac{d-p}{6d+2p}, & \text{otherwise} , \end{cases}$$

(ignoring lower order terms) that succeeds with high probability.

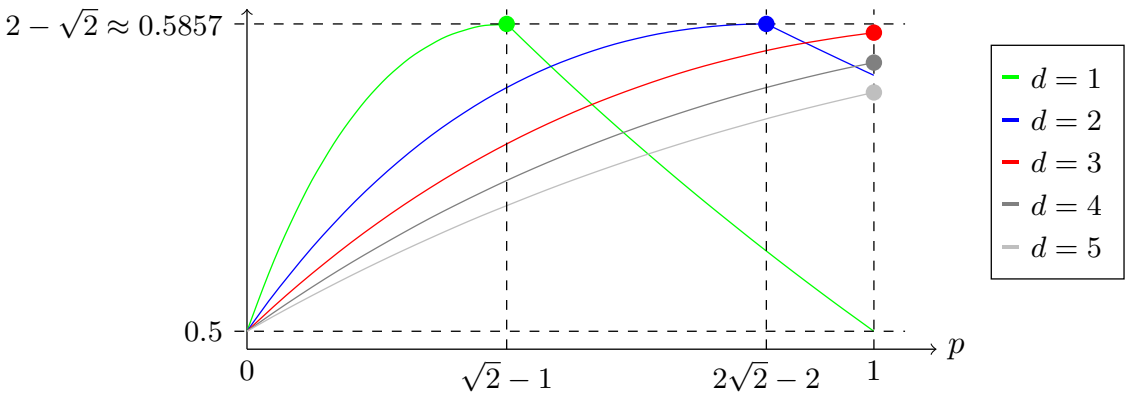


Figure 5.2: Approximation factors for different settings of d.

Interestingly, two parameter settings maximize the approximation factor in Theorem 5.2, achieving the ratio $2 - \sqrt{2}$ (see Figure 5.2). This is achieved by setting $d = 1$ and $p = \sqrt{2} - 1$ which recovers Konrad's algorithm, and by setting $d = 2$ and $p = 2\sqrt{2} - 2$ which gives a new algorithm. The setting $d = 3$ and $p = 1$ yields the slightly weaker bound $\frac{1}{2} + \frac{1}{12} \approx 0.5833$ and recovers Esfandiari et al.'s algorithm.

We also show that the analysis of our meta-algorithm is tight, by giving instances on which it does not perform better than the claimed bound (see Theorem 5.14).

Discussion. Our results demonstrate that new techniques are needed in order to improve on the $2-\sqrt{2}$ approximation factor. However, one may wonder whether $2-\sqrt{2}$ is the best approximation ratio achievable by the class of two-pass matching algorithms that solely computes a maximal matching in the first pass. As pointed out by Kapralov [Kap21], their techniques for establishing the $\frac{1}{1+\ln 2}$ lower bound for one-pass algorithms can probably also be applied to a construction by Huang et al. [HPT+19], which would then show that $2-\sqrt{2}$ is the best approximation factor achievable by one-pass semi-streaming algorithms for MBM. It is unclear whether a first-pass GREEDY matching could be embedded in the resulting construction without affecting its hardness; however, if possible, this would render Konrad's algorithm optimal for the considered class of two-pass streaming algorithms.

Independent and Concurrent Work. Independently and concurrently to our work, Assadi [Ass22] gave a limitation result on the approximation factor achievable by *arbitrary* two-pass semi-streaming algorithms for MBM, i.e., algorithms with no restrictions on how they operate in the first pass. From the lower bound perspective, this setting is substantially harder to work with than the setting considered in this chapter, where we assume that algorithms run GREEDY in the first pass. Assadi showed that no (arbitrary) two-pass semi-streaming algorithm for MBM has an approximation factor better than $\left(1-\Omega(\frac{\log \mathrm{RS}(n)}{\log n})\right)$, where $\mathrm{RS}(n)$ denotes the maximum number of disjoint induced matchings of size $\Omega(n)$ in an n-vertex graph. Determining $\mathrm{RS}(n)$ is a challenging open problem in combinatorics, and, currently, the best upper and lower bounds are still very far apart from each other: $n^{\Omega(\frac{1}{\log\log n})} \leq \mathrm{RS}(n) \leq n^{1-o(1)}$ [FLN+02, FHS17]. In the best case scenario, i.e., if $\mathrm{RS}(n)$ was indeed as large as $n^{1-o(1)}$, their result would imply that no two-pass semi-streaming algorithm can achieve a better than 0.98-approximation, and as long as $\mathrm{RS}(n) = n^{\Omega(1)}$, their result would rule out small constant factor approximations.

Developments since the Conference Version [KN21] of this chapter. Following the conference version of this chapter [KN21], progress has been made on both the algorithm and lower bound fronts for two-pass semi-streaming MBM:

- Bhattacharya et al. [BKSW23] have recently developed a two-pass semi-streaming algorithm that deterministically achieves a $2-\sqrt{2}-\epsilon$ approximation factor for small ϵ; however, it introduces a space dependency on $\frac{1}{\epsilon}$. Their algorithm computes a maximal matching M in the first pass, then computes b-matchings in the subgraphs $G_L = G[A(M) \cup \overline{B(M)}]$ and $G_R = G[\overline{A(M)} \cup B(M)]$, respectively, in the second pass. The b-matchings computed are such that the vertices matched by M can have at most b_{in}

incident edges and the ones unmatched by M can have at most b_{out} incident edges, where b_{in} and b_{out} are optimally chosen to achieve the approximation factor. This algorithm thus falls into the aforementioned class of algorithms that we study in this chapter.

- In Chapter 6 (and published in [KN24]), which is work that followed the work in this chapter, we give the first unconditional two-pass lower bound for semi-streaming MBM. Using almost entirely disjoint techniques from this chapter and completely different ideas from Assadi [Ass22], we show that any better than an (8/9)-approximation for MBM requires more than semi-streaming space.

Further Related Work. Besides two passes over the input, improvements over the Greedy algorithm can also be obtained under the assumption that the input stream is in random order. Assadi and Behnezhad [AB21] showed that an approximation factor of $\frac{2}{3} + \epsilon$ can be obtained, for some fixed small but constant $\epsilon > 0$, building on Bernstein's breakthrough result [Ber20], and improving on previous algorithms [Ber20, FHM+20, Kon18b, KMM12]. In insertion-deletion streams, where previously inserted edges may be deleted again, space $\Theta(n^{2-3\epsilon})$ is necessary [DK20] and sufficient [AS22] for computing a n^{ϵ}-approximation (see also [Kon15, CCE+16, AKLY16]).

Technical Changes since the Conference Version. In this chapter, we correct two inaccuracies that appear in the conference version of this chapter [KN21]. First, in the proof of Lemma 9 of [KN21], an independence requirement for applying Wald's equation was not established – this is now corrected with a simple fix. Second, we observe that a slightly different definition of the martingales used in Lemma 10 of [KN21] is required to establish the claimed result. Lemma 10 of [KN21] closely follows a proof of [Kon18b], which employs the same flawed definition. We provide a fix that corrects both [KN21] and [Kon18b].

Outline. We first give notation and definitions in section 5.2. Subsequently, we show in section 5.3 that every two-pass semi-streaming algorithm that solely runs Greedy in the first pass cannot have an approximation ratio of $\frac{2}{3} + \epsilon$, for any $\epsilon > 0$. Our main algorithmic result, i.e., the combination of subsampling and Greedy$_d$, is presented in section 5.4. Finally, we conclude in section 5.5.

5.2 Preliminaries

Let $G = (A, B, E)$ be a n-vertex bipartite graph with $V = A \cup B$. For $F \subseteq E$ and $v \in V$, we write $\deg_F(v)$ to denote the degree of vertex v in subgraph (A, B, F). For any $U \subseteq V$ and $F \subseteq E$, $U(F)$ denotes the set of vertices in U which are also endpoints of edges in F, and we denote its complement by $\overline{U(F)} = U \setminus U(F)$. We write $G[U]$ for the subgraph of G induced by U. For any edges $e, f \in E$, e is *incident* to f if they share an endpoint. We say

that e and f are *vertex-disjoint* if e is not incident to f. For any two sets X and Y, we define $X \oplus Y := (X \setminus Y) \cup (Y \setminus X)$ as their symmetric difference. Lastly, a matching M is called an *induced matching* if the edge set of $G[V(M)]$ is exactly M.

Wald's Equation. We will require the following version of *Wald's Equation* adapted from the book by Mitzenmacher and Upfal [MU05, Theorem 12.3] (the proof is given in subsection 5.2.1 for completeness) which relies on the notion of a random *stopping time*:

Definition 5.3 (Stopping Time)**.** A non-negative, integer-valued random variable T is a stopping time for the sequence of random variables $X_1, X_2, \ldots$ if the event $\{T = n\}$ depends only on $X_1, \ldots, X_n$. T is considered bounded if there exists a constant $c \in \mathbb{N}$ such that $T \leq c$.

Proposition 5.4 (Wald's Equation)**.** *Let $X_1, X_2, \ldots$ be a sequence of non-negative independent random variables where $\mathbb{E}[X_i] \leq \tau$, for every $i \geq 1$, and let T be a bounded random stopping time for the sequence. Then:*

$$\mathbb{E}[\sum_{i=1}^{T} X_i] \leq \tau \cdot \mathbb{E}[T].$$

5.2.1 Proof of Proposition 5.4

Following Mitzenmacher and Upfal [MU05, Theorem 12.3], we prove Proposition 5.4 using Doob's well-known *Optional Stopping Theorem* for martingales (see [Wil91, Theorem 10.10] for a proof).

Proposition 5.5 (Doob's Optional Stopping Theorem)**.** *Let $Z_1, Z_2, \ldots$ be a martingale and let T be a bounded stopping time for the sequence. Then:*

$$\mathbb{E}[Z_T] = \mathbb{E}[Z_1].$$

Proposition 5.4 (Wald's Equation)**.** *Let $X_1, X_2, \ldots$ be a sequence of non-negative independent random variables where $\mathbb{E}[X_i] \leq \tau$, for every $i \geq 1$, and let T be a bounded random stopping time for the sequence. Then:*

$$\mathbb{E}[\sum_{i=1}^{T} X_i] \leq \tau \cdot \mathbb{E}[T].$$

Proof. Let $Z_i = \sum_{j=1}^{i}(X_j - \mathbb{E}[X_j])$. Then,

$$\mathbb{E}[Z_1] = \mathbb{E}[X_1 - \mathbb{E}[X_1]] = \mathbb{E}[X_1] - \mathbb{E}[\mathbb{E}[X_1]] = \mathbb{E}[X_1] - \mathbb{E}[X_1] = 0\ , \tag{5.1}$$

and, using the independence of the variables $X_1, X_2, \ldots$, we obtain

$$\mathop{\mathbb{E}}_{X_{i+1}}[Z_{i+1} \mid X_1, \ldots, X_i] = \mathop{\mathbb{E}}_{X_{i+1}}[\sum_{j=1}^{i+1}(X_j - \mathbb{E}[X_j]) \mid X_1, \ldots, X_i]$$

$$= \mathop{\mathbb{E}}_{X_{i+1}}[\sum_{j=1}^{i}(X_j - \mathbb{E}[X_j]) \mid X_1, \ldots, X_i] + \mathop{\mathbb{E}}_{X_{i+1}}[X_{i+1} - \mathbb{E}[X_{i+1}] \mid X_1, \ldots, X_i]$$
$$= \sum_{j=1}^{i}(X_j - \mathbb{E}[X_j]) + \underbrace{\mathbb{E}[X_{i+1} - \mathbb{E}[X_{i+1}]]}_{=0} = Z_i \ .$$

Hence, the sequence $Z_1, Z_2, \ldots$ is a martingale w.r.t. the sequence $X_1, X_2, \ldots$. Since T is bounded, we can apply Proposition 5.5 and obtain $\mathbb{E}[Z_T] = \mathbb{E}[Z_1]$, which further yields $\mathbb{E}[Z_T] = 0$ by Equation 5.1. The result then follows since

$$0 = \mathbb{E}[Z_T] = \mathbb{E}[\sum_{j=1}^{T}(X_j - \mathbb{E}[X_j])] \geq \mathbb{E}[\sum_{j=1}^{T}(X_j - \tau)] = \mathbb{E}[\sum_{j=1}^{T} X_j] - \mathbb{E}[T \cdot \tau].$$

□

5.3 Lower Bound

We now prove that every two-pass streaming algorithm for MBM with approximation factor $\frac{2}{3} + \epsilon$, for any $\epsilon > 0$, that solely runs Greedy in the first pass requires space $n^{1+\Omega(\frac{1}{\log \log n})}$. To this end, we adapt the lower bound by Goel et al. [GKK12], which we discuss first.

5.3.1 Goel et al.'s Lower Bound for One-pass Algorithms

Goel et al.'s lower bound is proved in the *one-way two-party communication framework*. Two parties, denoted Alice and Bob, respectively hold subsets E_1 and E_2 of the input graph's edges. Alice sends a single message to Bob who, upon receipt, outputs a large matching. Goel et al. showed that there is a distribution λ over input graphs such that a message of length $n^{1+\Omega(\frac{1}{\log \log n})}$ is required by every deterministic communication protocol with constant distributional error over λ and approximation factor $\frac{2}{3} + \epsilon$, for any $\epsilon > 0$. A similar result then applies to randomized constant error protocols by Yao's Lemma [Yao77], and the well-known connection between streaming algorithms and one-way communication protocols allows us to translate this lower bound to a lower bound on the space requirements of constant error one-pass streaming algorithms.

Goel et al.'s distribution is based on the existence of dense Ruzsa-Szemerédi graphs:

Definition 5.6 (Ruzsa-Szemerédi Graph)**.** A bipartite graph $G = (A, B, E)$ is an (r, t)-*Ruzsa-Szemerédi* graph (RS-graph in short) if the edge set E can be partitioned into t disjoint matchings $M_1, M_2, \ldots, M_t$ such that, for every i, the following holds:

1. $|M_i| = r$; and
2. M_i is an *induced matching* in G.

Goel et al. give a construction for a family of $((\frac{1}{2}-\delta)\cdot n, n^{\Omega(\frac{1}{\log\log n})})$-RS-graphs, for any small constant $\delta > 0$, on $2n$ vertices (with $|A| = |B| = n$) that we will extend further below. Based on their construction, they then build a hard input distribution λ for the two-party communication setting which we describe in Figure 5.3.

1. Let $G^{RS} = (A, B, E)$ be an (r, t)-RS-graph with $|A| = |B| = N$, $r = (\frac{1}{2}-\delta)\cdot N$, for some $\delta > 0$, and $t = N^{\Omega(\frac{1}{\log\log N})}$ (as implied by the RS-graph construction of Goel et al.).
2. For every $i \in [t]$, let $\widehat{M}_i \subseteq M_i$ be a uniform random subset of size $(\frac{1}{2}-2\delta)\cdot N$ and let $E_1 = \cup_{i=1}^{t} \widehat{M}_i$.
3. Let X and Y each be disjoint sets of $(\frac{1}{2}+\delta)\cdot N$ vertices that are also disjoint from $A \cup B$. Choose uniformly at random a special index $s \in [t]$. Let M_X^* and M_Y^* be arbitrary perfect matchings between X and $\overline{B(M_s)}$, and Y and $\overline{A(M_s)}$, respectively. Then, let $E_2 = M_X^* \cup M_Y^*$.
4. Finally, let $G = (A \cup X, B \cup Y, E_1 \cup E_2)$ which has $n = (3+2\delta)\cdot N$ vertices.

Alice holds edges E_1 and Bob holds edges E_2.

Figure 5.3: Hard input distribution λ.

Observe that the graphs $G \sim \lambda$ are such that $\mu(G) \geq \frac{3}{2}N$; the matching $M_X^* \cup M_Y^* \cup \widehat{M_s}$ defined in Figure 5.3 is of this size. With this, Goel et al. prove the following hardness result:

Proposition 5.7 ([GKK12]). *For any small $\epsilon > 0$, every deterministic $(\frac{2}{3}+\epsilon)$-approximation one-way two-party communication protocol with constant distributional error over λ requires a message of size $n^{1+\Omega(\frac{1}{\log\log n})}$, where n is the number of vertices in the input graph.*

5.3.2 Our Lower Bound Construction

In the following, we extend Goel et al.'s lower bound to the two-pass situation where a Greedy matching is computed in the first pass. To this end, we need to augment Alice and Bob's inputs, as defined by distribution λ, by a maximal matching M in the input graph $G \sim \lambda$, which then results in a distribution λ^+. Observe that if we place the edges of M at the beginning of the input stream, then running Greedy in the first pass recovers exactly the matching M. Hence, when abstracting the second pass as a two-party communication problem, both Alice and Bob already know the matching M. Our main argument then is as follows: We will show that any two-party protocol under distribution λ^+ can also be used for solving the distribution λ with the same distributional error, message size, and similar approximation factor. The hardness of Proposition 5.7, therefore, carries over.

5.3.2.1 Ruzsa-Szemerédi Graphs with Near-Perfect Matchings

Adding a maximal matching M to Alice's and Bob's input requires care since we need to ensure that the hardness of the construction is preserved. Our construction requires that the underlying RS-graph contains a near-perfect matching, which is a property that is not immediate from Goel et al.'s RS-graph construction.

We therefore augment Goel et al.'s construction by complementing every induced matching, M_i, with a vertex-disjoint counterpart, M_i', without destroying the RS-graph properties. Then, since M_i and M_i' are vertex-disjoint, $M_i \cup M_i'$ constitutes a matching, and, since both M_i and M_i' each already match nearly half of the vertices, $M_i \cup M_i'$ constitutes a near-perfect matching in our family of RS-graphs.

We will now present Goel et al.'s RS-graph construction and then discuss how the additional matchings M_i' can be added to the construction.

Goel et al.'s Ruzsa-Szemerédi Graph Construction

For an integer $m > 0$, let $X = Y = [m^2]^m$ be the vertex sets (represented as discrete vector spaces) of a bipartite graph, and let $N = |X| = |Y| = m^{2m}$ denote their cardinalities. Observe that every m-coordinate vector in this space represents a single vertex. Every induced matching M_I of Goel et al.'s RS-graph construction is indexed by a subset of coordinates $I \subset [m]$ of size $\frac{\delta m}{6}$, for some small $\delta > 0$. Then, the edges M_I are defined by means of an identical colouring of the vertex sets X and Y (which depends on I), that we discuss first.

Colouring the Vertex Sets. Let $w = \frac{(2+\delta)m}{3}$. Then, define a partition of the natural numbers into groups of size w such that, for all $k \in \mathbb{N}_0$,

$$
\begin{aligned}
R_k &= \left[kw, kw + \frac{m}{3}\right) && \text{where } |R_k| = \frac{m}{3},\\
W_k &= \left[kw + \frac{m}{3}, kw + \frac{m}{3} + \frac{\delta m}{6}\right) && \text{where } |W_k| = \frac{\delta m}{6},\\
B_k &= \left[kw + \frac{m}{3} + \frac{\delta m}{6}, kw + \frac{2m}{3} + \frac{\delta m}{6}\right) && \text{where } |B_k| = \frac{m}{3},\\
W_k' &= \left[kw + \frac{2m}{3} + \frac{\delta m}{6}, (k+1)w\right) && \text{where } |W_k'| = \frac{\delta m}{6}.
\end{aligned}
$$

See Figure 5.4 for an illustration.

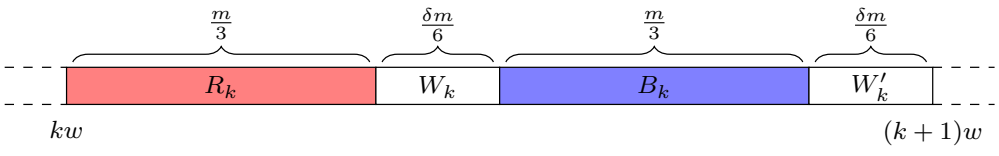


Figure 5.4: A single group of the partitioned number line of natural numbers.

Given a subset of coordinates I, let $L_s = \{\vec{x} \in [m^2]^m : \sum_{i \in I} x_i = s\}$ for any $s \in \mathbb{N}_0$ represent a layer of vectors in the vector space $[m^2]^m$ where the 1-norm of their subvectors[2] w.r.t. I is s. Next, we colour the vectors in each L_s either red if $s \in R_k$, blue if $s \in B_k$, or white if $s \in W_k \cup W'_k$, for any $k \in \mathbb{N}_0$. Doing this gives the following coloured strips for all $k \in \mathbb{N}_0$ (see Figure 5.5):

$$R(k) = \bigcup_{\forall s \in R_k} L_s, \quad W(k) = \bigcup_{\forall s \in W_k} L_s, \quad B(k) = \bigcup_{\forall s \in B_k} L_s \quad \text{and} \quad W'(k) = \bigcup_{\forall s \in W'_k} L_s.$$

Next, these strips are grouped together by colour, as follows:

$$R = \bigcup_{\forall k \in \mathbb{N}_0} R(k), \quad W = \bigcup_{\forall k \in \mathbb{N}_0} W(k), \quad B = \bigcup_{\forall k \in \mathbb{N}_0} B(k) \quad \text{and} \quad W' = \bigcup_{\forall k \in \mathbb{N}_0} W'(k).$$

Since $X = Y = [m^2]^m$, we now have that, given a specific subset of coordinates I, the vertices X and Y are coloured as follows: A vertex $\vec{z} \in X$ (resp. Y) is coloured red if $\vec{z} \in R$, blue if $\vec{z} \in B$, and white if $\vec{z} \in W \cup W'$. Let R^X be the red coloured vertices of X and define $B^X, W^X, W'^X, R^Y, B^Y, W^Y, W'^Y$ similarly.

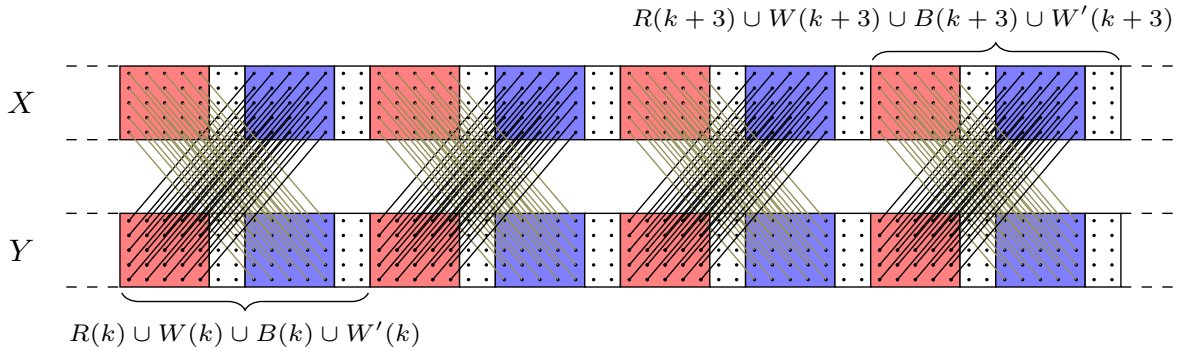


Figure 5.5: Illustration of the vertex colouring and induced matchings for a fixed I. The black edges are M_I and the gold ones are M'_I.

Definition of the Induced Matchings. Goel et al. construct the edges of the induced matching M_I by pairing every blue vertex $\vec{b} \in B^X$ with each coordinate greater than $\frac{2}{\delta} + 1$ to a red vertex $\vec{r} \in R^Y$, such that $\vec{r} = \vec{b} - (\frac{2}{\delta} + 1) \cdot \vec{1}_I$, where $\vec{1}_I$ is the characteristic vector of set I. They show that $|M_I| \geq (\frac{1}{2} - \delta) \cdot N - o(N)$. See Figure 5.5 for an illustration.

Observe that any two distinct indexing sets I and J produce their own vertex colourings and matchings M_I and M_J. Goel et al. prove that, as long as the index sets I and J have a sufficiently small intersection (at most $\frac{5\delta}{12} \cdot \frac{\delta m}{6}$), M_I and M_J are induced matchings w.r.t.

[2] A subvector in this context is the result of a trivial mapping of the vector to a lower dimensional subspace, i.e., considering only the coordinates indexed by I.

each other, i.e., each edge of M_I has at most one endpoint in $V(M_J)$ and vice versa. Hence, they show the existence of a large family $\mathcal{T}$ of $N^{\Omega(\frac{1}{\log\log N})}$ many indexing sets I whose pairwise intersections are of size at most $\frac{5\delta}{12} \cdot \frac{\delta m}{6}$. Then, the matchings of the RS-graph are identified as the matchings M_I, for every $I \in \mathcal{T}$.

Extending Goel et al.'s Construction

For every indexing set $I \in \mathcal{T}$ and respective matching M_I of Goel et al.'s construction, we symmetrically construct an additional matching M'_I by pairing every blue vertex $\vec{b} \in B^Y$ with each coordinate greater than $\frac{2}{\delta} + 1$ to a red vertex $\vec{r} \in R^X$ such that $\vec{r} = \vec{b} - (\frac{2}{\delta} + 1) \cdot \vec{1}_I$. Note that the roles of X and Y are reversed in the construction of M_I. See Figure 5.5 for an illustration.

We immediately see that, by virtue of being symmetrical, $|M'_I| = |M_I| (\geq (\frac{1}{2} - \delta) \cdot N - o(N))$. Furthermore, by construction, M'_I and M_I are vertex-disjoint matchings, hence $M_I \cup M'_I$ is a matching, and, taking their respective sizes into account, $M_I \cup M'_I$ is a near-perfect matching as required.

For any distinct $I, J \in \mathcal{T}$, we have that M_I and M_J are induced matchings w.r.t. each other; therefore, the symmetrical nature of our additional matchings implies the same for M'_I and M'_J. However, showing that M_I and M'_J are induced w.r.t. each other is not as obvious. Fortunately, a proof of this property is already implicit in Goel et al.'s analysis [GKK12] and, for completeness, we reproduce the decisive argument:

Lemma 5.8. *Given two distinct sets of indices I and J such that $|I \cap J| \leq \frac{5\delta}{12} \cdot \frac{\delta m}{6}$, no edge in M_I has both its endpoints in $V(M'_J)$, for any small enough $\delta > 0$.*

Proof. Let $\vec{b} \in B^X$ be matched to $\vec{r} \in R^Y$ by M_I, i.e., $\vec{b} - \vec{r} = (\frac{2}{\delta} + 1) \cdot \vec{1}_I$. If the edge $(\vec{b}, \vec{r})$ has both its vertices in $V(M'_J)$, then one endpoint is coloured blue and the other red in the colouring of X and Y with respect to J. Hence, $\vec{b}$ and $\vec{r}$ are separated by at least a single white strip (see Figure 5.5) implying that $|\sum_{j \in J} \left(\vec{b} - \vec{r}\right)_j| \geq \frac{\delta m}{6}$. On the other hand,

$$
\begin{aligned}
|\sum_{j \in J} \left(\vec{b} - \vec{r}\right)_j| &= \sum_{j \in J} \left(\left(\frac{2}{\delta} + 1\right) \cdot \vec{1}_I\right)_j && \text{(since } \vec{b} - \vec{r} = (\tfrac{2}{\delta} + 1) \cdot \vec{1}_I > 0\text{)} \\
&= \left(\frac{2}{\delta} + 1\right) \cdot |I \cap J| && \text{(since } \textstyle\sum_{j \in J} (\vec{1}_I)_j = |I \cap J|\text{)} \\
&\leq \left(\frac{5}{6} + \frac{5\delta}{12}\right) \cdot \frac{\delta m}{6}. && \text{(since } |I \cap J| \leq \tfrac{5\delta}{12} \cdot \tfrac{\delta m}{6}\text{)}
\end{aligned}
$$

Combining these bounds, we have that $\frac{\delta m}{6} \leq |\sum_{j \in J} (\vec{b} - \vec{r})_j| \leq (\frac{5}{6} + \frac{5\delta}{12}) \cdot \frac{\delta m}{6}$, which is a contradiction for small enough δ. □

We thus obtain the following theorem:

Theorem 5.9 (cf. Theorem 5)**.** *For any small enough constant $\delta > 0$, there exists a family of bipartite (r,t)-Ruzsa-Szemerédi graphs where $|A| = |B| = N$, $r = (\frac{1}{2} - \delta) \cdot N$, and $t = N^{\Omega(\frac{1}{\log \log N})}$ such that there are $\frac{1}{2} \cdot t$ edge-disjoint near-perfect matchings each of size $2 \cdot r$.*

5.3.2.2 Lower Bound Proof

Equipped with RS-graphs with near-perfect matchings and input distribution λ, we now define our hard input distribution λ^+, see Figure 5.6.

1. Let G^{RS} be a RS-graph as in Theorem 5.9. Fix some induced matching M and let $M \cup M'$ be its near-perfect matching of size $(1 - 2\delta) \cdot N$.
2. Let F be an arbitrary set of $2\delta N$ additional edges (not necessarily in G^{RS}) such that $P = M \cup M' \cup F$ matches all the vertices of G^{RS}.
3. Consider distribution λ constructed using RS-graph $G^{RS} \setminus (M \cup M')$.
4. For every $G = (A, B, E) \sim \lambda$, let $P_G = M \cup M' \cup (F \setminus E)$ (to avoid multi-edges) and add P_G to G to obtain the input graph G^+.

The edges $P \cup E_1$ are given to Alice and the edges $P \cup E_2$ are given to Bob (recall that E_1 and E_2 are defined in distribution λ).

Figure 5.6: Hard input distribution λ^+.

We are now ready to prove our main lower bound theorem:

Theorem 5.10. *For any $\epsilon > 0$, every deterministic $(\frac{2}{3} + \epsilon)$-approximation one-way communication protocol with constant distributional error over λ^+ for MBM requires a message of size $n^{1+\Omega(\frac{1}{\log \log n})}$, where n is the number of vertices in the input graph.*

Proof. Let γ^+ be a deterministic $(\frac{2}{3} + \epsilon)$-approximation protocol that solves distribution λ^+ with constant distributional error. Given γ^+, we will now define a protocol γ that solves distribution λ with the same communication cost, same error, and approximation ratio strictly better than $\frac{2}{3}$. Invoking Proposition 5.7 then proves our result.

The protocol γ is easy to obtain: Observe that P in distribution λ^+ is the same for every sampled input graph $G^+ \sim \lambda^+$. Hence, in protocol γ, Alice and Bob first make sure that the edges P are included in their inputs. This is achieved by Alice adding the edges $P \setminus E_1 = P_G$ to her input, and Bob adding the edges P to his input. In doing so, Alice and Bob's input is identically distributed to choosing an input graph G^+ from λ^+. Alice and Bob can, therefore, run protocol γ^+ which produces an output matching M^+_{out}. Bob then outputs the largest matching M_{out} among the edges $M^*_X \cup M^*_Y \cup (M^+_{\text{out}} \setminus P_G)$ as the output of the protocol γ.

Next, we will argue that $|M_{\text{out}}| \geq |M^+_{\text{out}}| - |F| = |M^+_{\text{out}}| - 2\delta N$. We can construct a matching $\tilde{M}$ of this size as follows: First, add every edge $e \in M^+_{\text{out}}$ that is not contained in P to $\tilde{M}$.

Second, for every edge $e \in M^+_{\text{out}} \cap (M \cup M')$, we insert the incident edge to e that is contained in $M^*_X \cup M^*_Y$ into $\tilde{M}$ (notice that these incident edges always exist except for edges from the special induced matching of distribution λ). This implies that $|M_{\text{out}}| \geq |\tilde{M}| \geq |M^+_{\text{out}}| - |F|$.

Now, we show that the approximation ratio is strictly greater than $\frac{2}{3}$. Recall that $\mu(G) \geq \frac{3}{2}N$ which implies that $N \leq \frac{2}{3}\mu(G)$, and since G is a subgraph of G^+, $\mu(G) \leq \mu(G^+)$. We know that γ^+ is a $(\frac{2}{3}+\epsilon)$-approximation protocol, so we have that $|M^+_{\text{out}}| \geq (\frac{2}{3}+\epsilon)\mu(G^+)$, and thus,

$$|M_{\text{out}}| \geq |M^+_{\text{out}}| - 2\delta N \geq (\frac{2}{3}+\epsilon)\mu(G^+) - 2\delta \cdot \frac{2}{3}\mu(G) \geq (\frac{2}{3}+\epsilon-\frac{4}{3}\delta)\mu(G) \ .$$

Hence, setting $\delta < \frac{3}{4}\epsilon$ in distribution λ yields a protocol with approximation ratio strictly above $\frac{2}{3}$. This, however, implies that γ requires a message of length $n^{1+\Omega(\frac{1}{\log\log n})}$ by Proposition 5.7, and since the message sent in γ and γ^+ is equivalent, the result follows. □

Applying Yao's Lemma and the usual connection between streaming algorithms and one-way communication protocols, we obtain our main lower bound result:

Theorem 5.1 (cf. Theorem 4). *For any $\epsilon > 0$, every (possibly randomized) two-pass streaming algorithm for MBM with approximation ratio $\frac{2}{3}+\epsilon$ that solely computes a* Greedy *matching in the first pass requires $n^{1+\Omega(\frac{1}{\log\log n})}$ space, where n is the number of vertices in the graph.*

5.4 Algorithm

In this section, we combine the subsampling approach as used by Konrad [Kon18b] and the semi-matching approach as used by Esfandiari et al. [EHM16] and Kale and Tirodkar [KT17] in order to find many disjoint 3-augmenting paths w.r.t. a maximal matching M. The algorithm is presented as Algorithm 4.

The input to Algorithm 4 is a stream of edges π of a bipartite graph $G = (A, B, E)$, a maximal matching M in G (e.g., computed in a first pass by Greedy), a sampling probability p, and an integral degree bound d. First, each edge of M is included in M' with probability p. Then, while processing the stream, degree-d-bounded semi-matchings S_L and S_R are computed using the algorithm Greedy_d (recall Figure 3). The algorithm then returns a largest subset of vertex-disjoint 3-augmenting paths $\mathcal{Q}$ in $M \cup S_L \cup S_R$. We can thus obtain a matching of size $|M| + |\mathcal{Q}|$.

Algorithm 4 Finding Augmenting Paths

Input: A stream of edges π of a bipartite graph $G = (A, B, E)$, a maximal matching M in G, $p \in (0, 1]$ and $d \in \mathbb{N}^+$.

1: Let $M' \subseteq M$ be a random subset such that $\forall e \in M$, $\Pr[e \in M'] = p$
2: Let $G'_L = G[A(M') \cup \overline{B(M)}]$ and $G'_R = G[\overline{A(M)} \cup B(M')]$
3: Denote by $\pi_{G'_L}$ and $\pi_{G'_R}$ the substream of π of edges of G'_L and G'_R, respectively
4: $S_L \leftarrow \textsc{Greedy}_d(\pi_{G'_L})$ such that $\deg_{S_L}(a) \leq 1$, for every $a \in A(M')$, and $\deg_{S_L}(b) \leq d$, for every $b \in \overline{B(M)}$
5: $S_R \leftarrow \textsc{Greedy}_d(\pi_{G'_R})$ such that $\deg_{S_R}(b) \leq 1$, for every $b \in B(M')$, and $\deg_{S_R}(a) \leq d$, for every $a \in \overline{A(M)}$
6: $\mathcal{P} \leftarrow \{(ab', ab, a'b) : \forall ab \in M' \text{ where } ab' \in S_L \text{ and } a'b \in S_R\}$
7: **return** A largest subset $\mathcal{Q} \subseteq \mathcal{P}$ of vertex-disjoint paths.

5.4.1 Analysis of Algorithm 4

The main task in analysing Algorithm 4 is to bound the sizes of S_L and S_R from below. A bound that holds in expectation for the case $d = 1$ was previously proved by Konrad et al. [KMM12], and a high probability result (for $d = 1$) was later obtained by Konrad [Kon18b]. We also first give a bound that holds in expectation (Lemma 5.11), which is achieved by extending the original proof by Konrad et al. [KMM12]. Our extension, however, is non-trivial as it requires a very different progress measure. Then, we obtain a high probability version in Lemma 5.12.

We also remark that Lemma 5.11 and Lemma 5.12 are stated in a more general context, however, it is not hard to see that they capture the situation of the computations of S_L and S_R in subgraphs G'_L and G'_R, respectively.

Lemma 5.11. *Let $G = (A, B, E)$ be a bipartite graph, π an arbitrarily ordered stream of its edges, $p \in (0, 1]$, and $d \in \mathbb{N}^+$. Let $A' \subseteq A$ be a random subset such that, for each $a \in A$, $\Pr[a \in A'] = p$ and let d be the degree bound of the B vertices. Let $H = G[A' \cup B]$ and denote by π_H the substream of π consisting of the edges in H. Then,*

$$\mathbb{E}_{A'}[|\textsc{Greedy}_d(\pi_H)|] \geq \frac{d}{d+p} \cdot p \cdot \mu(G).$$

Proof. Let M^* be a fixed maximum matching in G and let $M^*_H := \{ab \in M^* : a \in A'\}$ be the subset of its edges incident to A'.

Game Setup. We consider the following game: On the selection of an edge by $\textsc{Greedy}_d(\pi_H)$, the edge *attacks* the (at most two) incident edges of M^*_H, dealing damage to them. Initially, the damage of every edge in M^*_H is 0 and, over the course of the algorithm, each of them accumulates damage. A damage strictly less than 1 means that the edge could still be selected by the algorithm; otherwise, it already has been or no longer can be selected.

Denote by $e_1, e_2, \ldots$ the sequence of edges S selected by $\textsc{Greedy}_d(\pi_H)$ and let $S_i = \{e_1, \ldots, e_i\}$ be the first i edges selected. The way damage is dealt by edge $e_i = ab$ is covered by the following two rules:

R1 If there is an edge $ab' \in M_H^*$ then attack edge ab' by adding $1 - \frac{\deg_{S_{i-1}}(b')}{d}$ damage to it.

R2 If there is an edge $a'b \in M_H^*$ such that $a' \notin A(S_i)$ then attack edge $a'b$ by adding $\frac{1}{d}$ damage to it.

Note that if $ab \in M_H^*$ then only R1 applies since $a = a' \in A(S_i)$ necessarily holds. Next, we argue that, by the end of the algorithm, every edge in M_H^* is dealt a damage of 1; hence, the total damage dealt by the selected edges S will always total $|M_H^*|$.

Consider first the case where an edge $e^* \in M_H^*$ is dealt damage according to R1 by a selected edge e_i. Then, we have that $A(e^*) \in S_j$ for all $j \geq i$, which immediately implies that e^* no longer accumulates damage according to R2 and, since any subsequent edge e_j selected by $\textsc{Greedy}_d(\pi_H)$ can not be not incident to $A(e^*)$, e^* will no longer accumulate damage according to R1 either, halting any damage done to it. Notice then that prior to selecting edge e_i, e^* could have only accumulated damage according to R2 and, thus, had a total accumulated damage of $\frac{1}{d} \cdot \deg_{S_{i-1}}(B(e^*)) \leq 1$. This means that the selected edge e_i, which deals $1 - \frac{1}{d} \cdot \deg_{S_{i-1}}(B(e^*))$ damage, indeed maxes out the damage accumulated by e^* to 1. Finally, in the case where e^* does not ever accumulate damage according to R1 (i.e., $\deg_S(A(e^*)) = 0$), we necessarily have that $e^* \notin S$ and that $\deg_S(B(e^*)) = d$. Therefore, the total damage accumulated by e^* is still maxed out at 1.

Applying Wald's Equation. Recall that $e_1, e_2, \ldots$ is the sequence of edges that are selected by $\textsc{Greedy}_d(\pi_H)$ and let $e_i = \perp$ for all $i > |\textsc{Greedy}_d(\pi_H)|$. We want to bound the expected size of the matching computed for any run of the algorithm. In essence, we do this using the fact that, by the end of any run of the algorithm, exactly $|M_H^*|$ damage is dealt by the selected edges. To that end, we bound the expected damage dealt by each selected edge, then, using the fact that $|\textsc{Greedy}_d(\pi_H)|$ edges are selected by the algorithm, we get a bound on the total expected damage dealt, which then implies the result. This final step, however, requires Wald's Equation since the number of edges selected is a random variable.

Let Y_i be the damage dealt by e_i conditioned on the state of the algorithm prior to processing edge e_i. This natural conditioning allows us to consider the damage dealt w.r.t. any run of the algorithm. As such, we trivially have that the random variables $Y_1, Y_2, \ldots$ are independent. Let T be the smallest i such that $\sum_{j=1}^{i} Y_j = |M_H^*|$ holds. Observe that $T \leq |\textsc{Greedy}_d(\pi_H)| \leq |\pi_H| \leq \binom{n}{2} \in \mathbb{N}$ is a bounded random stopping time (see Definition 5.3). To apply the version of Wald's Equation presented in Proposition 5.4, it remains to find a value τ such that $\mathbb{E}[Y_i] \leq \tau$ holds for all $i \geq 1$.

Observe that the maximum damage an edge selected by $\textsc{Greedy}_d(\pi_H)$ can inflict is $1+\frac{1}{d}$ (applying R1 and R2 simultaneously). Therefore, the damage Y_i dealt by any edge e_i is either $0, \frac{1}{d}, \frac{2}{d}, \ldots, 1$ or $1+\frac{1}{d}$. Hence, we obtain the following:

$$\mathbb{E}[Y_i] \leq \Pr[Y_i \leq 1] \cdot 1 + \underbrace{\Pr\left[Y_i = 1+\frac{1}{d}\right]}_{=q} \cdot (1+\frac{1}{d}) = (1-q)\cdot 1 + q\cdot(1+\frac{1}{d}) = 1+\frac{q}{d}.$$

It remains to bound the probability that edge $e_i = ab$ deals the largest possible damage, i.e., $\Pr[Y_i = 1+\frac{1}{d}](= q)$. By definition of the game, if the event $Y_i = 1+\frac{1}{d}$ occurs then there exists an edge $a'b \in M^*_H$ such that $a' \notin A(S_i)$. However, observe that since $a' \notin A(S_i)$, the random choice as to whether $a' \in A'$ and thus whether $a'b \in M^*_H$ had not needed to occur yet (principle of deferred decision). Hence, we obtain

$$\Pr[Y_i = 1+\frac{1}{d}] \leq \Pr[a' \in A'] = p,$$

which implies $\mathbb{E}[Y_i] \leq 1+\frac{p}{d}$.

Having shown that $Y_1, Y_2, \ldots$ are independent, T is bounded, and $\mathbb{E}[Y_i] \leq 1+\frac{p}{d}$ for all $i \geq 1$, we can apply Proposition 5.4 (Wald's Equation) to obtain $\mathbb{E}[\sum_{j=1}^{T} Y_i] \leq (1+\frac{p}{d})\mathbb{E}[T]$. Finally, since $\mathbb{E}[\sum_{j=1}^{T} Y_i] = \mathbb{E}[|M^*_H|] = p\cdot\mu(G)$ and $T \leq |\textsc{Greedy}_d(\pi_H)|$, it follows that

$$\mathbb{E}[\sum_{j=1}^{T} Y_i] = p\cdot\mu(G) \leq (1+\frac{p}{d})\cdot\mathbb{E}[T] \leq (1+\frac{p}{d})\cdot\mathbb{E}[|\textsc{Greedy}_d(\pi_H)|],$$

which implies the result. □

In subsection 5.4.3, we follow a similar approach as Konrad [Kon18b] to strengthen Lemma 5.11 and obtain the following high probability result:

Lemma 5.12. *Let $G=(A,B,E)$ be a bipartite graph, π be any arbitrary stream of its edges, $p \in (0,1]$ and $d \in \mathbb{N}^+$. Let $A' \subseteq A$ be a random subset such that, for each $a \in A$, $\Pr[a \in A'] = p$. Let d be the degree bound of the B vertices and let $H = G[A' \cup B]$. Then, the following holds with probability at least $1-2\mu(G)^{-18}$:*

$$|\textsc{Greedy}_d(\pi_H)| \geq \frac{d}{d+p}\cdot p\cdot\mu(G) - o(\mu(G)).$$

Equipped with Lemma 5.12, we are now ready to bound the number of augmenting paths found by Algorithm 4.

Lemma 5.13. *Suppose that $|M| = (\frac{1}{2}+\epsilon)\mu(G)$ and let Q be the set of vertex-disjoint 3-augmenting paths found by Algorithm 4. Then, with probability at least $1-\mu(G)^{-16}$, we have that*

$$|\mathcal{Q}| \geq \left(\frac{1-2\epsilon}{d+p} - \frac{1+2\epsilon}{2d}\right)\cdot p\cdot\mu(G) - o(\mu(G)).$$

Proof. Let M^* be a fixed maximum matching in G. In this proof, we will refer to the quantities used by Algorithm 4. First, using a Chernoff bound for independent Poisson trials, we see that

$$|M'| = p \cdot |M| \pm O(\sqrt{|M| \ln |M|}) \tag{5.2}$$

with probability at least $1 - |M|^{-C}$ for an arbitrarily large constant C.

Consider the subgraphs $G_L = G[A(M) \cup \overline{B(M)}]$ and $G_R = G[\overline{A(M)} \cup B(M)]$. $M \oplus M^*$ contains $(\frac{1}{2} - \epsilon)\mu(G)$ vertex-disjoint augmenting paths where each path starts and ends with an edge in $G_L \cup G_R$. This implies that

$$\mu(G_L) + \mu(G_R) \geq 2(\frac{1}{2} - \epsilon)\mu(G) = (1 - 2\epsilon)\mu(G). \tag{5.3}$$

Following Konrad [Kon18b], we will argue next that

$$|\mathcal{P}| \geq |S_L| + |S_R| - |M'|. \tag{5.4}$$

Observe that there are $|M'| - |S_L|$ vertices of $|M'|$ that are not incident to an edge in S_L, and similarly, $|M'| - |S_R|$ vertices of $|M'|$ that are not incident to an edge in S_R. Hence, there are at least $|M'| - (|M'| - |S_L|) - (|M'| - |S_R|) = |S_L| + |S_R| - |M'|$ edges of $|M'|$ that are incident to both an edge from S_L and S_R. We thus obtain that there are at least $|\mathcal{P}| \geq |S_L| + |S_R| - |M'|$ 3-augmenting paths.

Next, Esfandiari et al. (Lemma 6 in [EHM16]) consider a similar structure to $\mathcal{P}$ and argue that there is at least a $(1/d)$-portion of augmenting paths in $\mathcal{P}$ that are vertex-disjoint; hence,

$$\begin{aligned}
|\mathcal{Q}| &\geq \frac{1}{d} \cdot |\mathcal{P}| \\
&\geq \frac{1}{d} \cdot (|S_L| + |S_R| - |M'|) && \text{(by Equation 5.4)} \\
&\geq \frac{1}{d} \cdot \Big(|S_L| + |S_R| - p \cdot (\frac{1}{2} + \epsilon)\mu(G) - o(\mu(G))\Big) && \text{(by Equation 5.2 w.h.p.)} \\
&\geq \frac{1}{d} \cdot \Big(\frac{d}{d+p} \cdot p \cdot (\mu(G_L) + \mu(G_R)) - p \cdot (\frac{1}{2} + \epsilon)\mu(G) - o(\mu(G))\Big) \\
& && \text{(by Lemma 5.12 w.h.p.)} \\
&\geq \frac{1}{d} \cdot \Big(\frac{d}{d+p} \cdot p \cdot (1 - 2\epsilon)\mu(G) - p \cdot (\frac{1}{2} + \epsilon)\mu(G) - o(\mu(G))\Big) && \text{(by Equation 5.3)} \\
&= \Big(\frac{1 - 2\epsilon}{d+p} - \frac{1 + 2\epsilon}{2d}\Big) \cdot p \cdot \mu(G) - o(\mu(G)).
\end{aligned}$$

Finally, by the union bound, the error is bounded by $|M|^{-C} + 4\mu(G)^{-18} \leq \mu(G)^{-16}$. □

We are now ready to state our main algorithmic result:

Theorem 5.2 (cf. Theorem 3). *For every $p \in (0, 1]$ and every integral $d \geq 1$, there is a two-pass semi-streaming algorithm for MBM with approximation factor*

$$\begin{cases} \frac{1}{2} + (\frac{1}{d+p} - \frac{1}{2d}) \cdot p - o(1), & \text{if } p \leq d(\sqrt{2}-1) \\ \frac{1}{2} + \frac{d-p}{6d+2p} - o(1), & \text{otherwise} , \end{cases}$$

that succeeds with high probability (in $\mu(G)$, where G is the input graph). The settings ($d = 1, p = \sqrt{2}-1$) and ($d = 2, p = 2(\sqrt{2}-1)$) maximize the approximation factor to $2-\sqrt{2}-o(1)$.

Proof. Let M be a maximal matching such that $|M| = (\frac{1}{2} + \epsilon)\mu(G)$, for some $0 \leq \epsilon \leq \frac{1}{2}$ and some bipartite graph $G = (A, B, E)$ with a stream π of its edges. Let $\mathcal{Q}$ be the set of disjoint augmenting paths found by Algorithm 4 on input π, M, p and d. Then, augmenting M with $\mathcal{Q}$ yields a matching of size $|M| + |\mathcal{Q}|$. By Lemma 5.13, the following inequality holds with high probability:

$$|M| + |\mathcal{Q}| \geq (\frac{1}{2} + \underbrace{\epsilon}_{(a)})\mu(G) + \underbrace{\max\left\{\left(\frac{1-2\epsilon}{d+p} - \frac{1+2\epsilon}{2d}\right) \cdot p, 0\right\}}_{(b)} \cdot \mu(G) - o(\mu(G)). \tag{5.5}$$

The goal now is to minimize the RHS of Equation 5.5 w.r.t. ϵ. Observe that we only need to consider the relevant parts labelled (a) and (b), where (a) is an increasing function of ϵ and (b) is a decreasing one. We first make two key observations:

- $\epsilon = 0$ minimizes (a); and
- $\epsilon \geq \frac{d-p}{6d+2p}$ minimizes (b) since $(b) = 0$ due to the max function.

Now, for any fixed choices of p and d, if (a) increases at a greater rate than the decrease of (b), then minimizing (a) minimizes Equation 5.5, i.e., $\epsilon = 0$, otherwise the minimizing choice of (b) that also minimizes (a) minimizes it, i.e., $\epsilon = \frac{d-p}{6d+2p}$. It can be shown that the turning point between these cases is for the choices where $p = d(\sqrt{2}-1)$.

Overall, we have the following two cases:

1. If $p \leq d(\sqrt{2}-1)$ then $\epsilon = 0$ minimizes the RHS of Equation 5.5, and we obtain the claimed bound by plugging the value $\epsilon = 0$ into the inequality.
2. If $p \geq d(\sqrt{2}-1)$ (only possible if $d \in \{1, 2\}$) then $\epsilon = \frac{d-p}{6d+2p}$ minimizes the RHS of Equation 5.5, and we obtain the claimed bound by plugging the value $\epsilon = \frac{d-p}{6d+2p}$ into the inequality.

It can be seen that, for a fixed d, the best approximation factor is obtained if $p = \min\{d\sqrt{2} - d, 1\}$, and the values $d \in \{1, 2\}$ yield the claimed bound of $2 - \sqrt{2} - o(1)$ (see Figure 5.2 in section 5.1). □

5.4.2 Optimality of the Analysis

We will show now that our analysis of Algorithm 4 is best possible. To this end, we define a worst-case input graph G in Figure 5.7 and prove in Theorem 5.14 that Algorithm 4 does not perform better on G than predicted by our analysis. See Figure 5.8 for an illustration.

1. Let $A_{\text{in}} = \{a^1_{\text{in}}, a^2_{\text{in}}, \ldots, a^N_{\text{in}}\}$, $A_{\text{out}} = \{a^1_{\text{out}}, \ldots, a^N_{\text{out}}\}$, $B_{\text{in}} = \{b^1_{\text{in}}, \ldots, b^N_{\text{in}}\}$, and $B_{\text{out}} = \{b^1_{\text{out}}, \ldots, b^N_{\text{out}}\}$ be sets of vertices, for some integer N.
2. Let $M = \{a^i_{\text{in}} b^i_{\text{in}} \ : \ 1 \leq i \leq N\}$ be a perfect matching between A_{in} and B_{in}. Let $G_L = (A_{\text{in}}, B_{\text{out}}, E_L)$ be a semi-complete graph such that $a^i_{\text{in}} b^j_{\text{out}} \in E_L \Leftrightarrow i \geq j$, and let $G_R = (A_{\text{out}}, B_{\text{in}}, E_R)$ be a semi-complete graph such that $a^i_{\text{out}} b^j_{\text{in}} \in E_R \Leftrightarrow i \geq j$.
3. Our bipartite hard instance graph is defined as $G = (A_{\text{in}} \cup A_{\text{out}}, B_{\text{in}} \cup B_{\text{out}}, M \cup E_L \cup E_R)$ and has $n = 4N$ vertices.
4. Finally, let π be a stream of its edges where the edges of M arrive first followed by the edges E_L and E_R. The edges in E_L are ordered so that $a^i_{\text{in}} b^j_{\text{out}}$ arrives before $a^{i'}_{\text{in}} b^{j'}_{\text{out}}$ only if $i > i'$, or $i = i'$ and $j < j'$. Similarly, the edges in E_R are ordered so that $a^i_{\text{out}} b^j_{\text{in}}$ arrives before $a^{i'}_{\text{out}} b^{j'}_{\text{in}}$ only if $i > i'$, or $i = i'$ and $j < j'$.

Figure 5.7: Hard input instance G for Algorithm 4.

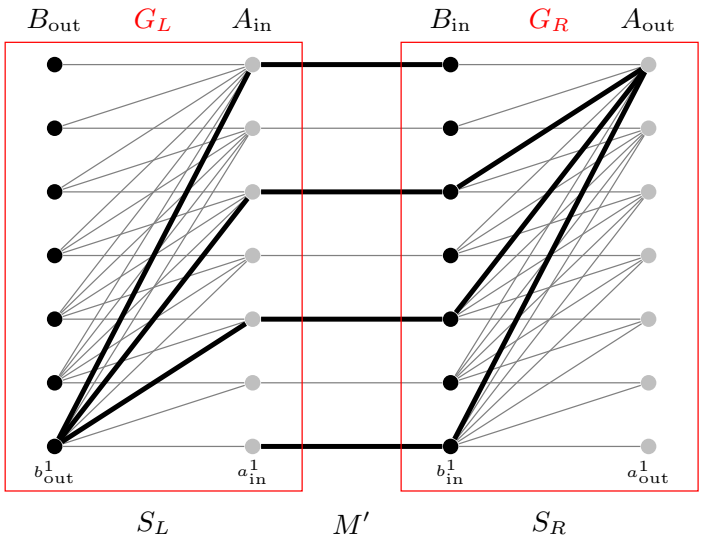


Figure 5.8: Algorithm 4 on a hard input instance G with $N = 7$, $d = 3$ and $p = 0.5$. The grey edges represent the edges of G while the thick black edges represent the edges of M', S_L and S_R.

Observe that M is a maximal matching in G, and if we run Greedy in the first pass on π then M would be returned. Let $M^*_L = \{a^i_{\text{in}} b^i_{\text{out}} \ : \ 1 \leq i \leq N\}$ and $M^*_R = \{a^i_{\text{out}} b^i_{\text{in}} \ : \ 1 \leq i \leq N\}$. Then, M^*_L is a perfect matching in G_L, M^*_R is a perfect matching in G_R, and $M^*_L \cup M^*_R$ is a

perfect matching in G.

Theorem 5.14. *Algorithm 4 with parameters $d \geq 1$ and $0 < p \leq 1$ on input G received via stream π and maximal matching M finds at most*

$$\left(\left(\frac{1}{d+p} - \frac{1}{2d}\right) \cdot p + o(1)\right) \mu(G)$$

augmenting paths with high probability. This renders our analysis of Algorithm 4 best possible when $p \leq d\sqrt{2} - d$.

Proof. In this proof, we will refer to the quantities used by Algorithm 4, that is, M' (the edges of M subsampled with probability p), S_L and S_R.

In our subsequent arguments, we implicitly condition on the event that the statement in the following claim holds:

Claim 5.15. *With high probability, for every pair $i, j \in [N]$ with $i \leq j$, we have*

$$|\{a^k_{in} b^k_{in} \in M' \mid i \leq k \leq j\}| = p \cdot (j - i) \pm o(N) \ .$$

Proof. This claim is easy to prove. Indeed, for any fixed $i, j \in [N]$ with $i \leq j$, the statement above follows directly from the Chernoff bound. Using the union bound over all pairs $i, j \in [N]$, the claim follows. □

Let $A'_{\text{in}} = A(M')$ and let $B'_{\text{in}} = B(M')$. We will first argue that, for two different vertices $a^i_{\text{in}}, a^j_{\text{in}} \in A'_{\text{in}}$ with $i < j$, if $a^i_{\text{in}} \in A(S_L)$ then $a^j_{\text{in}} \in A(S_L)$ also holds. Indeed, suppose that this was not the case. Let b^k_{out} be the partner of a^i_{in} in S_L. Observe that the edges $a^i_{\text{in}} b^k_{\text{out}}, a^j_{\text{in}} b^k_{\text{out}} \in E_L$, and, in particular, the edge $a^j_{\text{in}} b^k_{\text{out}}$ arrives before the edge $a^i_{\text{in}} b^k_{\text{out}}$ in π. Hence, edge $a^j_{\text{in}} b^k_{\text{out}}$ would have been selected, a contradiction. A similar argument holds for vertices $b^i_{\text{out}}, b^j_{\text{out}} \in B_{\text{out}}$ with $i > j$; if $\deg_{S_L}(b^i_{\text{out}}) = d$ then $\deg_{S_L}(b^j_{\text{out}}) = d$.

Let $i_{\min}$ be the smallest index such that $a^{i_{\min}}_{\text{in}} \in A(S_L)$. We will now argue that $i_{\min} \geq \frac{pN}{p+d} - o(N)$. Observe that the vertices A'_{in} are matched in order from the largest to smallest index, and each matched vertex in A'_{in} is matched only once. The vertices in B_{out} are matched from the smallest to largest index, and each matched vertex is matched d times (except possibly the last such matched vertex). Consider the last edge $a^{i_{\min}}_{\text{in}} b^q_{\text{out}}$ inserted into S_L. Then, $q \leq i_{\min}$, and, thus, $|B(S_L)| \leq i_{\min}$. By Claim 5.15, we have $|A(S_L)| \geq p \cdot (N - i_{\min}) - o(N)$ with high probability. Since $A(S_L)$ is matched to $B(S_L)$ in S_L, and each B-vertex is matched at most d times, we obtain $|A(S_L)| \leq d \cdot |B(S_L)|$, and, hence:

$$p \cdot (N - i_{\min}) - o(N) \leq |A(S_L)| \leq d \cdot |B(S_L)| \leq d \cdot i_{\min} \ ,$$

which implies $i_{\min} \geq \frac{pN}{p+d} - o(N)$.

Let $i_{\max}$ be the largest index such that $b_{\text{in}}^{i_{\max}} \in B(S_R)$. Using a similar argument as above, we see that $i_{\max} \leq \frac{dN}{p+d} + o(N)$.

Let $M'' = \{a_{\text{in}}^i b_{\text{in}}^i \in M' : i_{\min} \leq i \leq i_{\max}\}$ be the subset of augmentable edges, i.e., edges for which there exists a left wing in S_L and a right wing in S_R. Then, by Claim 5.15, we have

$$|M''| \leq p \cdot (i_{\max} - i_{\min}) + o(N) \leq \frac{p(d-p)N}{p+d} + o(N) \ .$$

All but constantly many vertices in $A(M'')$ share the same neighbour in S_L with $d-1$ other vertices of $A(M'')$. Hence, at most a d-fraction (plus up to the constantly many exceptions, which disappear in the $o(N)$ term) of M'' can be augmented simultaneously. Using $N = \frac{1}{2}\mu(G)$, we obtain the following bound on the number of edges that can be augmented simultaneously:

$$\frac{1}{d}|M''| \leq \frac{1}{d}\left(\frac{p(d-p)N}{p+d} + o(N)\right) = \left(\left(\frac{1}{d+p} - \frac{1}{2d}\right) \cdot p + o(1)\right)\mu(G) \ .$$

□

5.4.3 High probability version of Lemma 5.11

Following the work by Konrad [Kon18b], we use tail inequalities for martingales to strengthen Lemma 5.11 and give a high probability result. The proof of Lemma 5.12 uses *Azuma-Hoeffding's Inequality* [MU05, Theorem 12.4]:

Lemma 5.16 (Azuma-Hoeffding's Inequality)**.** *Let $Z_0, Z_1, ..., Z_n$ be a martingale such that $|Z_{i+1} - Z_i| \leq c_i$. Then, $\forall t \geq 0$ and any $\lambda > 0$,*

$$Pr\left[|Z_t - Z_0| \geq \lambda\right] \leq 2\exp\left(\frac{-\lambda^2}{2\sum_{i=0}^{t-1} c_i^2}\right).$$

To apply a combinatorial result previously shown by Konrad [Kon18b, Lemma 5] (restated here as Lemma 5.18), we first prove a useful result (Lemma 5.17) that allows us to transform any instance of GREEDY$_d$ into an instance of GREEDY that returns the same number of edges.

Lemma 5.17. *For any graph $G = (A, B, E)$ with an arbitrary stream of its edges π and a vertex-induced subgraph $H = G[A' \cup B]$ where $A' \subseteq A$, there exists a graph $G^+ = (A, B^+, E^+)$ with a stream of its edges π^+ and a vertex-induced subgraph $H^+ = G^+[A' \cup B^+]$ such that, for any B-vertex degree bound $d \in \mathbb{N}^+$,*

$$|\textsc{Greedy}_d(\pi_H)| = |\textsc{Greedy}(\pi^+_{H^+})|.$$

Proof. Consider any stream of edges π of a bipartite graph $G = (A, B, E)$. We first construct a graph $G^+ = (A, B^+, E^+)$ from graph G such that $b_{(i,1)}, \ldots, b_{(i,d)} \in B^+$ for every $b_i \in B$, and $(a, b_{(i,1)}), \ldots, (a, b_{(i,d)}) \in E^+$ for every $(a, b_i) \in E$. In essence, G^+ is the graph G with every

B-vertex and its incident edges replaced by d copies. The stream of its edges π^+ can then be constructed from π where every edge $(a, b_i) \in \pi$ is replaced with d copies $(a, b_{(i,1)}), \ldots, (a, b_{(i,d)})$. Note that the stream π^+ maintains the same ordering as stream π, that is, if the edge (a, b_i) arrives before the edge (a, b_j) in the stream π, then the edges $(a, b_{(i,1)}), \ldots, (a, b_{(i,d)})$ arrive before the edges $(a, b_{(j,1)}), \ldots, (a, b_{(j,d)})$ in the stream π^+. Now, we have that the vertex-induced subgraph $H = G[A' \cup B]$ has a stream of edges π_H and, by following an identical construction, it is easy to see that the graph and stream we obtain is the vertex-induced subgraph $H^+ = G^+[A' \cup B^+]$ and stream $\pi^+_{H^+}$.

Let $S = \textsc{Greedy}_d(\pi_H)$ and $M = \textsc{Greedy}(\pi^+_{H^+})$. We argue that, for every $a \in A'$, $\deg_S(a) = \deg_M(a)$ and, for every $b_i \in B$, $\deg_S(b_i) = \deg_M(b_{(i,1)}) + \cdots + \deg_M(b_{(i,d)})$ where $b_{(i,1)}, \ldots, b_{(i,d)} \in B^+$. This can be shown by induction on the arrival of each edge (a, b_i) of the stream π_H w.r.t. the arrival of the last edge among the corresponding copies $(a, b_{(i,1)}), \ldots, (a, b_{(i,d)})$ in the stream $\pi^+_{H^+}$. Then, the result holds by observing that the number of edges incident to A-vertices (or B-vertices) in S and M are equivalent. □

Lemma 5.18 ([Kon18b, Lemma 5])**.** *Let $G = (A, B, E)$ be a bipartite graph, π be any arbitrary stream of its edges and $p \in (0, 1]$. Then, for any $v \in A \cup B$ and $G \backslash v = G[(A \cup B) \backslash \{v\}]$ the following holds:*

$$0 \leq |\textsc{Greedy}(\pi)| - |\textsc{Greedy}(\pi_{G \backslash v})| \leq 1.$$

We now use these results to strengthen Lemma 5.11 to a high probability result:

Lemma 5.12. *Let $G = (A, B, E)$ be a bipartite graph, π be any arbitrary stream of its edges, $p \in (0, 1]$ and $d \in \mathbb{N}^+$. Let $A' \subseteq A$ be a random subset such that, for each $a \in A$, $\Pr[a \in A'] = p$. Let d be the degree bound of the B vertices and let $H = G[A' \cup B]$. Then, the following holds with probability at least $1 - 2\mu(G)^{-18}$:*

$$|\textsc{Greedy}_d(\pi_H)| \geq \frac{d}{d+p} \cdot p \cdot \mu(G) - o(\mu(G)).$$

Proof. Define $Y := |\textsc{Greedy}_d(\pi_H)|$. By applying Lemma 5.17 to graph G and stream π, we get graphs $G^+ = (A, B^+, E^+)$ and $H^+ = G^+[A' \cup B^+]$, and streams π^+ and $\pi^+_{H^+}$, respectively, such that $Y = |\textsc{Greedy}(\pi^+_{H^+})|$, thus putting us in exactly the same situation as Konrad's work [Kon18b, Theorem 6]. In the remainder of the proof, we only consider the setting on stream π^+ and, for simplicity, we denote $\textsc{Greedy}(\pi^+_{H^+})$ as $\textsc{Greedy}(H^+)$.

The proof follows by considering the information revealed about A' as $\textsc{Greedy}(H^+)$ processes the stream. Let e be an edge in the stream π^+ and let $M_{\neg e}$ be the matching computed before processing edge e. Edge e reveals whether $A(e) \in A'$ or $A(e) \notin A'$ only if e is the first edge in the stream incident to $A(e)$ that, upon arrival, has its corresponding B^+-vertex unmatched. In particular, if e is added to the matching then $A(e) \in A'$, otherwise $A(e) \notin A'$.

Let A_i represent the first i vertex reveals about A' and let $A_{>i} = A \backslash A_i$ represent the remaining unrevealed vertices. Let M_i represent the matching computed after the i^{th} vertex reveal. Then, $G^+_{>i} = G^+[A_{>i} \cup \overline{B^+(M_i)}]$ is the vertex-induced residual subgraph after the i^{th} reveal whose earliest arriving edge w.r.t. stream π^+ triggers the $(i+1)^{\text{th}}$ reveal about A'. Let $i^* \leq |A|$ be the final vertex reveal where $A_{>i^*}$ may still contain unrevealed vertices. However, since $G^+_{>i^*}$ must not contain any edges, the outcomes of these vertices do not affect the size of the matching and we have that

$$M_{i^*} = \textsc{Greedy}(H^+). \tag{5.6}$$

Define $X_i = (a_i, I(a_i))$, for all $i \leq i^*$, to be a pair of random variables that represent the i^{th} vertex reveal where a_i is the identity of the A-vertex and $I(a_i) = 1$ if $a_i \in A'$ and $I(a_i) = 0$ otherwise. Since $A_{>i^*}$ contains $|A| - i^*$ unrevealed vertices, we arbitrarily define $X_{i^*+1}, \ldots, X_{|A|}$ to be the remaining reveals of the vertices in $A_{>i^*}$. Now, we have that $X_1, \ldots, X_{|A|}$ are, in essence, the indicator random variables for whether each A-vertex is in A'; however, the order in which they are revealed is dependent on the specific random choice of A'.

With the goal of showing that Y is concentrated around its expectation, we define its corresponding Doob Martingale w.r.t. the sequence of random variables $X_1, \ldots, X_{|A|}$ where, for all $i = 0, \ldots, |A|$,

$$Z_i := \mathop{\mathbb{E}}_{X_{i+1}, \ldots, X_{|A|}}[Y | X_1, \ldots, X_i].$$

By Lemma 5.11, we have that $Z_0 = \mathbb{E}_{X_1, \ldots, X_{|A|}}[Y] = \mathbb{E}_{A'}[Y] \geq \frac{d}{d+p} \cdot p \cdot \mu(G)$ and, by Equation 5.6, we have that $Z_{i^*} = \cdots = Z_{|A|} = Y$. As such, we prove the result by showing that any deviation of $Z_{|A|}$ from Z_0 is small with high probability. We claim first that, for all $i < |A| < n$, the absolute difference $|Z_{i+1} - Z_i|$ is bounded above by 1. Then, by Lemma 5.16, we have that

$$\Pr\left[|Z_{|A|} - Z_0| \geq 6\sqrt{n \ln \mu(G)}\right] \leq 2\mu(G)^{-18},$$

where $|Z_{|A|} - Z_0| = |Y - \mathbb{E}[Y]|$.

Note that this proof only needs to give concentration on input graphs G where $\mu(G) = \omega(\sqrt{n \ln n})$ since, for graphs where $\mu(G) = O(\sqrt{n \ln n})$, we can run the one-pass algorithm by Chitnis et al. [CCE+16] that computes an exact maximum matching in $\mu(G)^2$ space, i.e., semi-streaming space. Therefore, unless this one-pass algorithm exceeds semi-streaming space, i.e., $\mu(G) = \omega(\sqrt{n \operatorname{polylog} n})$, we will have learned a maximum matching in the graph, which is even better, and in the case that memory is exceeded, we know that $\mu(G) = \omega(\sqrt{n \ln n})$ – giving us concentration by the analysis.

Bounding the Difference. We now show that $|Z_{i+1} - Z_i| \leq 1$ for all $i < |A|$. Firstly, for all $i \geq i^*$, we have from Equation 5.6 that $|Z_{i+1} - Z_i| = 0$, and it remains only to consider $i < i^*$.

Let $A'_i = A_i \cap A'$ and $A'_{>i} = A_{>i} \cap A'$, for all $i \leq i^*$. Let $H_i^+ = G^+[A'_i \cup B^+(M_i)]$ and $H_{>i}^+ = G^+[A'_{>i} \cup \overline{B^+(M_i)}]$. It is easy to see that $M_i = \textsc{Greedy}(H_i^+) \subseteq M_{i^*}$ is a perfect matching in H_i^+ and, since $H_{>i}^+$ is the residual subgraph induced by the vertices of H^+ unmatched by M_i, we have that $M_{i^*} \backslash M_i = \textsc{Greedy}(H_{>i}^+)$. Hence, for any $i \leq i^*$, it follows from Equation 5.6 that

$$\begin{aligned} Y &= |\textsc{Greedy}(H^+)| = |M_{i^*}| = |\textsc{Greedy}(H_i^+)| + |\textsc{Greedy}(H_{>i}^+)| \\ &= |M_i| + |\textsc{Greedy}(H_{>i}^+)| = |A'_i| + |\textsc{Greedy}(H_{>i}^+)| \\ &= \sum_{j=1}^{i} I(a_j) + |\textsc{Greedy}(H_{>i}^+)|. \end{aligned} \tag{5.7}$$

From the way we have defined the Doob Martingale, we have that

$$\begin{aligned} Z_{i+1} - Z_i &= \mathop{\mathbb{E}}_{X_{i+2},\ldots,X_{|A|}}[Y \mid X_1, \ldots, X_{i+1}] - \mathop{\mathbb{E}}_{X_{i+1},\ldots,X_{|A|}}[Y \mid X_1, \ldots, X_i] \\ &= \mathop{\mathbb{E}}_{X_{i+2},\ldots,X_{|A|}}\left[Y - \mathop{\mathbb{E}}_{X_{i+1}}[Y] \mid X_1, \ldots, X_{i+1}\right] && \text{(by linearity of expectation)} \\ &= \mathop{\mathbb{E}}_{X_{i+2},\ldots,X_{|A|}}\Bigg[\sum_{j=1}^{i+1} I(a_j) + |\textsc{Greedy}(H_{>i+1}^+)| \\ &\quad - \mathop{\mathbb{E}}_{X_{i+1}}\left[\sum_{j=1}^{i+1} I(a_j) + |\textsc{Greedy}(H_{>i+1}^+)|\right] \mid X_1, \ldots, X_{i+1}\Bigg] && \text{(by Equation 5.7)} \\ &= \mathop{\mathbb{E}}_{X_{i+2},\ldots,X_{|A|}}\left[I(a_{i+1}) + |\textsc{Greedy}(H_{>i+1}^+)| - \mathop{\mathbb{E}}_{X_{i+1}}\left[I(a_{i+1}) + |\textsc{Greedy}(H_{>i+1}^+)|\right] \mid X_{i+1}\right]. \end{aligned} \tag{5.8}$$

where the final equality holds since, for any fixed sequence of outcomes $X_1, \ldots, X_i$, i.e., a fixed choice of A'_i consistent with π^+, $\sum_{j=1}^{i} I(a_j) = |A'_i|$ is also fixed and independent of X_{i+1}. Furthermore, we drop the dependence on $X_1, \ldots, X_i$ since $I(a_{i+1})$ and the residual subgraph $H_{>i+1}^+$ are only dependent on X_{i+1} and the sequence of outcomes $X_{i+2}, \ldots, X_{|A|}$, i.e., the random choice of $A'_{>i+1}$.

Overall, to bound $Z_{i+1} - Z_i$, we give a stronger argument than required by Equation 5.8. We show that the largest difference between $I(a_{i+1}) + |\textsc{Greedy}(H_{>i+1}^+)|$ for any pair of outcomes of X_{i+1} is at most 1, for any fixed choice of $A'_{>i+1}$. This is indeed a stronger argument since Equation 5.8 is the largest difference, on average over all choices of $A'_{>i+1}$, between any outcome of $I(a_{i+1}) + |\textsc{Greedy}(H_{>i+1}^+)|$ and its average outcome w.r.t. X_{i+1}.

Let e_{i+1} be the first arriving edge in $G_{>i}^+$. We have that, for any fixed A'_i, the graph $G_{>i}^+$ is also fixed and thus the identity of the subsequent reveal $a_{i+1} = A(e_{i+1})$ is fixed. This implies that X_{i+1} only has two possible outcomes, neither of which affect the choice of $A'_{>i+1}$; in particular, $X_{i+1} = (A(e_{i+1}), 1)$ if e_{i+1} is added to the matching M_i and $X_{i+1} = (A(e_{i+1}), 0)$

if it is not. In the case where $X_{i+1} = (A(e_{i+1}), 1)$, we have that $H^+_{>i+1} = G^+[A'_{>i+1} \cup \overline{B^+(M_i)} \backslash B^+(e_{i+1})]$, whereas in the case where $X_{i+1} = (A(e_{i+1}), 0)$, we have that $H^+_{>i+1} = G^+[A'_{>i+1} \cup \overline{B^+(M_i)}]$. Since $H^+_{>i+1}$ differs by a single vertex in either case, Lemma 5.18 implies that, when $X_{i+1} = (A(e_{i+1}), 1)$ and e_{i+1} is added to M_i, $\textsc{Greedy}(H^+_{>i+1})$ contributes at most one fewer edge to the final matching M_{i^*} as compared to when $X_{i+1} = (A(e_{i+1}), 0)$ and e_{i+1} is not added to M_i. Therefore, the difference between $I(a_{i+1}) + |\textsc{Greedy}(H^+_{>i+1})|$ in either case of X_{i+1} is at most 1 and the claim flows since, from Equation 5.8, we have that

$$-1 \leq Z_{i+1} - Z_i \leq 1.$$

□

5.5 Conclusion

In this chapter, we studied the class of two-pass semi-streaming algorithms for MBM that solely compute a Greedy matching in the first pass. We showed that algorithms of this class cannot have an approximation ratio of $\frac{2}{3}+\epsilon$, for any $\epsilon > 0$. We also combined the two dominant techniques that have previously been used for designing such algorithms and discovered another algorithm that matches the state-of-the-art approximation factor of $2 - \sqrt{2} \approx 0.58578$.

We conclude with two open problems. First, we are particularly interested in whether there exists a one-pass semi-streaming algorithm that is able to augment a maximal matching so as to yield an approximation ratio above $2 - \sqrt{2}$. Second, is there a two-pass semi-streaming algorithm for MBM that improves on the approximation factor of $2-\sqrt{2}$ and operates differently in the first pass to the class of algorithms considered in this chapter?

6 Arbitrary Approximate MBM

This chapter has been published as the following work:

- [KN24]: Christian Konrad and Kheeran K. Naidu. An Unconditional Lower Bound for Two-Pass Streaming Algorithms for Maximum Matching Approximation. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024*.

Abstract

In this chapter, we give the first unconditional space lower bound for two-pass streaming algorithms for Maximum Bipartite Matching approximation. We show that every randomized two-pass streaming algorithm that computes a $(\frac{8}{9}+\epsilon)$-approximation to Maximum Bipartite Matching, for any constant $\epsilon > 0$, requires space $n^{1+\Omega(\frac{1}{(\log\log n)^2})}$, where n is the number of vertices of the input graph.

Previously, only a conditional lower bound by Assadi [SODA'22] was known that relates the quality of their lower bound to the maximum density of Ruzsa-Szemerédi graphs (RS-graphs) with matchings of linear sizes. In the best case, i.e., if very dense RS-graphs with linear-sized matchings exist, their lower bound rules out approximation ratios above 0.98, however, with current knowledge, only approximation factors of $1 - o(1)$ are ruled out.

Our lower bound makes use of the information cost trade-off of the Index problem in the two-party communication setting established by Jain et al. [JACM'09]. To the best of our knowledge, our work is the first that exploits this trade-off result in the context of lower bounds for multi-pass graph streaming algorithms.

6.1 Introduction

In the *streaming model of computation*, an algorithm processes its input sequentially by performing one or multiple passes over the input data. The objective is to design algorithms that use as little space as possible, in particular, space that is sublinear in the size of the input. This model addresses the challenge of processing massive data sets that exceed the size of the local memory of modern computers, and, due to the abundance of Big Data, the streaming model has received significant attention for more than two decades [Mut05, McG14].

In this chapter, we consider streaming algorithms for the Maximum Matching (MM) problem and its bipartite version, the Maximum Bipartite Matching (MBM) problem. Given an input graph $G = (V, E)$ with $n = |V|$, a streaming algorithm processes a stream or sequence of the edges of G in one or multiple passes and outputs an approximation to MM (or MBM) after having processed the data. Most fundamental graph problems have been studied in the streaming model, however, MM has received the most attention and can be considered the canonical graph problem for the study of the streaming model, see [ABKL23, ADKN23, KNS23, AS22, AJJ+22, Ass22, ALT21, KN21, CKP+21, AB21, BdBM21, Kap21, AR20, Ber20, DK20, KK20, FHM+20] for very recent works on this topic.

Semi-streaming algorithms [FKM+04], i.e., streaming algorithms for graph problems that use space $O(n \operatorname{poly} \log n)$, have received particular attention since many graph problems cannot be solved with less space (see, for example, [SW15]). Regarding MM, it was observed as early as in 2004 [FKM+04] that the Greedy matching algorithm that inserts an incoming edge into an initially empty matching if possible constitutes a one-pass 1/2-approximation semi-streaming algorithm, and the question as to whether there is a one-pass semi-streaming algorithm with improved approximation guarantee is one of the most significant open problems in the area. Progress has been made on the lower bound front, narrowing the scope for improvements in the approximation factor to the range $(1/2, \frac{1}{1+\ln(2)}] \approx (1/2, 0.59]$ [Kap21] (see also [Kap13, GKK12]).

The fact that no improvements in the one-pass model have been established in the last two decades motivates the study of slightly relaxed models. The two-pass semi-streaming setting is one possible relaxation that allows for the design of improved algorithms. The currently best two-pass algorithm for MBM is by Konrad [Kon18b] (see also [KN21]) who gave a $(2-\sqrt{2})$-approximation, improving over [KT17, EHM16, KMM12]. For general graphs, Azarmehr et al. [ABR24] very recently also established a $(2-\sqrt{2})$-approximation two-pass algorithm, matching the best result for bipartite graphs and improving over [KMM12, KT17]. On the lower bound front, Konrad and Naidu [KN21] showed that two-pass semi-streaming algorithms that solely run Greedy in the first pass cannot have an approximation factor better than 2/3. Assadi's work [Ass22] is the only lower bound result known that applies to arbitrary two-pass streaming algorithms, however, the quality of their lower bound depends on the maximum density of

Ruzsa-Szemerédi graphs with matchings of linear sizes, or equivalently, the maximum number of induced matchings of linear sizes that can be packed into a bipartite graph. The currently best construction yields $n^{\Omega(1/\log\log n)}$ matchings [FLN+02] (see also [GKK12]), while it is only known that packing more than $n/2^{O(\log^*(n))}$ matchings is impossible [FHS17]. In the best case, Assadi's lower bound rules out an approximation factor of 0.98, and, when employing the currently best construction with $n^{\Omega(1/\log\log n)}$ matchings, only $(1-o(1))$-approximation factors are ruled out.

6.1.1 Our Result

In this chapter, we substantially improve over Assadi's two-pass lower bound result and prove the following theorem:

Theorem 6.1 (cf. Theorem 6)**.** *For any constant $\epsilon > 0$, any constant-error two-pass $(\frac{8}{9}+\epsilon)$-approximation streaming algorithm for MBM requires $n^{1+\Omega(\frac{1}{(\log\log n)^2})}$ bits of space, where n is the number of vertices of the input graph.*

Our lower bound result implies that there is no two-pass semi-streaming algorithm for MBM with approximation factor $\frac{8}{9}+\epsilon$, for any constant $\epsilon > 0$. Our result improves over Assadi's lower bound in that it is unconditional, and it is substantially stronger since Assadi's result only rules out an approximation factor of 0.98 in the best case.

6.1.2 Techniques

The Index Problem

At the heart of our argument lies a reduction to the Index problem in the two-party communication setting. While many graph streaming lower bounds exploit this connection, our technique appears to be the first that exploits the information cost trade-off result for interactive communication protocols for Index as established by Jain et al. [JRS09]. We refer the reader to [KN96] and [Bra17] for excellent introductions to communication complexity and to information theory applied to proving lower bounds for interactive communication.

The Index_n problem is a two-party communication problem where the first party, denoted Alice, holds an n-bit string $Y \in \{0,1\}^n$, and the second party, denoted Bob, holds an index $J \in [n]$. The two parties communicate according to some pre-specified protocol π_{I} until one of the parties outputs the solution bit $Y[J]$. We ambiguously denote the transcript of the protocol, i.e., the collection of messages exchanged, also by π_{I}. The two parties are allowed to use both public (shared) and private randomness, and the objective is to produce a valid solution with probability 2/3.

If communication is restricted to a single message from Alice to Bob, and Bob needs to produce the output, then it is easy to show that, in an information theoretic sense where Y and J are distributed independently and uniformly at random according to distribution $\mathcal{D}_{\mathsf{I}}$, Alice's message must reveal $\Omega(n)$ bits about Alice's input to Bob, i.e., $\mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(Y;\pi_{\mathsf{I}} \mid J, R_{\mathsf{I}}) = \Omega(n)$, where R_{I} are the public random bits used by protocol π_{I} and $\mathcal{I}_{\mathcal{D}}(A;B \mid C)$ denotes the conditional mutual information of A and B conditioned on C distributed according to $\mathcal{D}$. The quantity $\mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(Y;\pi_{\mathsf{I}} \mid J, R_{\mathsf{I}})$ is typically referred to as the *(internal) information cost* of Alice of the protocol π_{I} and abbreviated by $\mathsf{ICost}^{A}_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}})$, and, similarly, the quantity $\mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(J;\pi_{\mathsf{I}} \mid Y, R_{\mathsf{I}})$ is referred to as the (internal) information cost of Bob of the protocol π_{I} and abbreviated by $\mathsf{ICost}^{B}_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}})$. Conversely, if Bob sends a single message to Alice, then sending J allows Alice to solve the problem, and, thus, we obtain $\mathsf{ICost}^{B}_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) = O(\log n)$. It is easy to see that information cost is a lower bound on communication cost, hence, the established bounds also apply to the number of bits that need to be communicated. If interactive communication is allowed, i.e., Alice and Bob communicate in multiple rounds, then the following information cost trade-off result can be established:

Proposition 6.2 (Information cost trade-off for Index_n)**.** *Let $b \leq O(\log n)$ be an integer, and let $\delta < \frac{1}{2}$ be a positive constant. Any δ-error protocol π_{I} for Index_n is such that:*

1. *Either $\mathsf{ICost}^{A}_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) = \mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(Y;\pi_{\mathsf{I}} \mid J, R_{\mathsf{I}}) \geq \frac{n}{2^{O(b)}}$, or*
2. *$\mathsf{ICost}^{B}_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) = \mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(J;\pi_{\mathsf{I}} \mid Y, R_{\mathsf{I}}) \geq b$,*

where $\mathcal{D}_{\mathsf{I}}$ denotes the uniform distribution and R_{I} is the public randomness used by π_{I}.

This result was first proved by Jain et al. [JRS09] in the context of 'privacy loss' in quantum communication; however, the version we present and require is in the context of (internal) information cost in classical communication. For completeness, in Appendix A.1, we show how this exact statement can be achieved using the information cost trade-off result for *augmented index verification* [CK11] (see also [CCKM13, JN14]).

The information cost trade-off result for Index is tight[1]. It shows that every bit revealed about Bob's input exponentially reduces the amount of information that a protocol must reveal about Alice's input. We exploit this information cost trade-off to prove a lower bound on a new communication problem that we denote $\mathsf{HiddenStrings}$.

The HiddenStrings Problem

The $\mathsf{HiddenStrings}$ problem defined below captures the hardness of our two-pass streaming lower bound for MM. $\mathsf{HiddenStrings}$ is a three-party communication game, and we denote the

[1] E.g., first Bob sends the b most significant bits of J to Alice. Alice then sends the $n/2^b$ positions of Y that are consistent with the b bits received from Bob back to Bob.

involved parties by Alice, Bob, and Charlie. Communication occurs in order and in two rounds, and Charlie produces the result at the end of the second round:

$$\underbrace{\text{Alice} \xrightarrow{\pi^{A1}_{\mathsf{HS}}} \text{Bob} \xrightarrow{\pi^{B1}_{\mathsf{HS}}} \text{Charlie} \xrightarrow{\pi^{C1}_{\mathsf{HS}}}}_{\text{Round 1}} \underbrace{\text{Alice} \xrightarrow{\pi^{A2}_{\mathsf{HS}}} \text{Bob} \xrightarrow{\pi^{B2}_{\mathsf{HS}}} \text{Charlie}}_{\text{Round 2}} \rightarrow \text{result} \ .$$

The problem is parametrized by four integers t_g, r_g, t_ℓ, r_ℓ (g and ℓ refer to *global* and *local*, respectively, whose meanings will become clear later). For each $i \in [t_g], j \in [r_g], k \in [t_\ell]$, Alice holds a bit-string $X[i,j,k] \in \{0,1\}^{r_\ell}$. For each $i \in [t_g]$ and $j \in [r_g]$, Bob holds an index $K[i,j] \in [t_\ell]$. Charlie holds an index $I^* \in [t_g]$. The goal is to output the strings $X[I^*, j, K[I^*, j]]$, for every $j \in [r_g]$. See Figure 6.1 for an illustration.

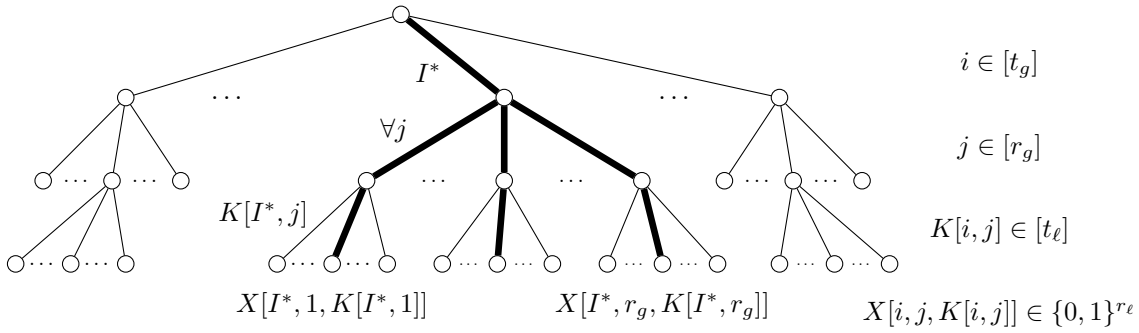


Figure 6.1: Illustration of the HiddenStrings problem. Alice's input strings $X[i,j,k]$ are arranged in a tree structure where the index i corresponds to the second level, j to the third level, and k to the fourth level. The strings $X[i,j,k]$ are located at the leaves of the tree. The objective is to output the strings located at the leaves in the subtree consisting of the bold edges, i.e., the strings $X[I^*, j, K[I^*, j]]$, for all $j \in [r_g]$. We see that Charlie's input I^* corresponds to a single edge between the root and the level two node. Then, all edges between the level two node and its incident level three nodes are selected. Finally, Bob's input, i.e., the values $K[i,j]$, select a single level four node for each selected level three node.

The HiddenStrings problem suggests two baseline protocols. In the first protocol, in the first round, only Charlie forwards the index I^*. Then, Alice forwards all strings $X[I^*, j, k]$, for every $j \in [r_g]$ and $k \in [t_\ell]$, in the second round, which yields a protocol with communication cost $\tilde{\mathrm{O}}(r_g \cdot t_\ell \cdot r_\ell)$. In the second protocol, Bob and Charlie forward their entire inputs in the first round, which allows Alice to compute and forward the solution in the second round. This yields a protocol with communication cost $\tilde{\mathrm{O}}(t_g \cdot r_g + r_g \cdot r_\ell)$.

We prove that these protocols are best possible in the following sense:

Theorem 6.3 (cf. Theorem 7)**.** *For any constant $\delta < \frac{1}{2}$, any δ-error two-round protocol for* HiddenStrings *requires $\Omega(\min\{t_g \cdot r_g, r_g \cdot t_\ell \cdot r_\ell\})$ bits of communication.*

This result is achieved via a reduction to Index_{t_ℓ}, i.e., we show that a protocol π_{HS} for HiddenStrings can be used to obtain a two-round protocol π_{I} that solves Index_{t_ℓ}. To this end, in

π_{I}, Alice and Bob first use public randomness to sample special indices $I^* \in [t_g]$, $J^* \in [r_g]$, and $\varphi^* \in [r_\ell]$. Then, given an instance $(Y \in \{0,1\}^{t_\ell}, J \in [t_\ell])$ of Index_{t_ℓ}, for every $k \in [t_\ell]$, the value $Y[k]$ is embedded as $X[I^*, J^*, k][\varphi^*] = Y[k]$ in the HiddenStrings instance. Alice generates all other entries of $X[i,j,k]$ at random using private randomness. Furthermore, Bob uses private randomness to randomly generate the values $K[i,j]$, for every i,j, but then updates $K[I^*, J^*]$ as $K[I^*, J^*] = J$. Charlie's input is set to I^*. Alice and Bob then simulate π_{HS}: Second-round Alice in π_{I} simulates first-round Charlie of π_{HS} and second-round Bob in π_{I} simulates second-round Charlie in π_{HS}, which is possible since I^* is generated using public randomness in π_{I}.

The correctness of this simulation is immediate; the solution to HiddenStrings must contain the solution to the embedded Index_{t_ℓ} instance. The crux of the analysis is to show that, if π_{HS} communicates at most $O(t_g \cdot r_g)$ bits in the first round, then the protocol π_{I} reveals only $O(1)$ bits about the position $J(= K[I^*, J^*])$, or, in other words, we have $\mathsf{ICost}^B_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) = O(1)$. The information cost trade-off for Index (Proposition 6.2) then implies that π_{I} needs to reveal $\Omega(t_\ell)$ bit about Y, i.e., $\mathsf{ICost}^A_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) = \Omega(t_\ell)$. However, since the embedding of Y is at random locations $X[I^*, J^*, k][\varphi^*]$, for every $k \in [t_\ell]$, and J^* and φ^* are chosen at random and independently, a direct-sum-like argument allows us to conclude that, in the second round of π_{HS}, Alice must essentially send all the strings $X[I^*, j, k]$, for every $j \in [r_g]$ and $k \in [t_\ell]$, to Bob, which amounts to $\Omega(r_g \cdot t_\ell \cdot r_\ell)$ bits. This establishes the result.

From Matching to HiddenStrings

To obtain our lower bound for MBM, we show that any two-pass streaming algorithm $\mathcal{A}$ for MBM with approximation factor at least $\frac{8}{9} + \epsilon$ can be used to solve HiddenStrings. To this end, given an input instance (X, K, I^*) of HiddenStrings, Alice, Bob, and Charlie encode the instance into a bipartite input graph on which they simulate a run of $\mathcal{A}$ by forwarding $\mathcal{A}$'s memory state. The input graph is constructed via the help of *Ruszsa-Szemerédi graphs*:

Definition 6.4 (Ruzsa-Szemerédi Graph)**.** A bipartite graph $G = (U, V, E)$ with $|U| = |V|$ is a (r,t)-Ruzsa-Szemerédi graph (RS-graph) if its edge set can be partitioned into t induced matchings $M_1, M_2, \ldots, M_t$, each of size r.

Let G^{RS}_g be a (r_g, t_g)-RS-graph, which we refer to as the global RS-graph. We replace each vertex of G^{RS}_g by separate vertex groups and each edge of G^{RS}_g by local (r_ℓ, t_ℓ)-RS-graphs on the respective vertex groups. Observe that the resulting graph has $t_g \cdot r_g \cdot t_\ell$ *local matchings*, i.e., matchings contained in the local RS-graphs, each consisting of r_ℓ edges. Recall that Alice's input X in HiddenStrings consists of $t_g \cdot r_g \cdot t_\ell$ bit-strings $X[i,j,k] \in \{0,1\}^{r_\ell}$. We can thus establish a one-to-one correspondence between the bits in X and the edges of the local RS-graphs: Depending on the value of the bits $X[i,j,k]$, Alice possibly deletes the edges that correspond to these bits.

Bob and Charlie hold additional *selector edges* that render some of the local induced matchings important in that a better than $\frac{8}{9}$-approximation to MBM must necessarily contain some edges of these important induced matchings. Consider Charlie first, and recall that Charlie holds a single index $I^* \in [t_g]$. Charlie introduces additional edges that render only the I^*th global induced matching of G_g^{RS} important, or, in other words, only those local RS-graphs that correspond to the edges of the I^*th global induced matching are important. Regarding Bob, for each local RS-graph, Bob adds additional selector edges that render a single induced matching of the local RS-graph important. Recall that Bob holds indices $K[i,j] \in [t_\ell]$, for every $i \in [t_g]$ and $j \in [r_g]$, which can be interpreted as selecting the $K[i,j]$th induced matching of the local RS-graph that corresponds to the jth edge of the ith global induced matching.

With the established correspondence, we see that level one in Figure 6.1 corresponds to selecting one global induced matching, level two corresponds to the edges of the global induced matchings (which in turn correspond each to one local RS-graph), level three corresponds to selecting one local induced matching for each local RS-graph, and the bits in the respective strings $X[i,j,k]$ correspond to the edges of the local induced matchings. Combining Charlie's and Bob's edges, we see that only the r_g local RS-graphs that correspond to the edges of one global induced matching are important, and, for each of these local RS-graphs, only the existence of the edges within a single local induced matching is important. Identifying these edges yields information about the bits $X[I^*, j, K[I^*, j]]$, for every $j \in [r_g]$, and thus solves HiddenStrings.

The implementation of this idea requires solving multiple technical challenges. First, since the algorithm $\mathcal{A}$ only produces an $(\frac{8}{9}+\epsilon)$-approximation to MBM, we only learn a $\Theta(\epsilon)$-fraction of the required bits for solving HiddenStrings from a run of $\mathcal{A}$. We thus run $\mathcal{A}$ often enough in parallel ($\Theta(\frac{1}{\epsilon} \cdot \operatorname{polylog} n)$ times is sufficient) and permute the input bits in X in each run in a suitable manner so that all bits have a similar probability of being learnt. Furthermore, we require that the bit strings $X[i,j,k]$ behave like uniform random bits. To this end, using public randomness, we XOR each of these bits with a uniform random bit to establish the required property. We then observe that we cannot directly translate these uniform random bits into edges being present (if the bit is 1) or absent (if the bit is 0) since this would imply that, in expectation, only half of the edges of a local induced matching are contained in the instance. We, however, require that all but a small linear fraction of edges are included in the matching to obtain the approximation lower bound of $\frac{8}{9}$. This is achieved by a probabilistic edge inclusion process: Edges whose corresponding bit in (the now uniform random) X is 0 are introduced with probability $1 - 2\epsilon$, while edges whose corresponding bit in X is 1 are always introduced. Hence, the probability of observing an edge whose corresponding bit has value 0 is slightly lower than the probability of observing an edge whose corresponding bit is 1. The number of parallel runs is chosen so that we can distinguish between the two cases for all relevant bits, which allows us to solve HiddenStrings.

6.1.3 Comparison to Goel et al. [GKK12] and to Assadi [Ass22]

As previously discussed, our work significantly improves over the conditional two-pass lower bound by Assadi [Ass22]. Both Assadi's work and ours make use of the lower bound construction by Goel et al. [GKK12], which rules out the existence of better than $\frac{2}{3}$-approximation one-pass semi-streaming algorithms for MBM, but in completely different ways. We first discuss Goel et al.'s construction and then compare Assadi's techniques to ours.

Goel et al. were the first to employ RS-graphs for obtaining space lower bounds for streaming algorithms for MBM. The key idea is that an RS-graph can be used to hide an important induced matching whose edges are required in any large matching. Their lower bound is proved in the one-way two-party communication setting. In their construction, Alice is given the edges of an RS-graph, and Bob is given selector edges that render one of the induced matchings important. Goel et al. showed that, in any one-way protocol, if Bob is able to learn even just a constant fraction of the edges of the important induced matching then Alice must essentially send the entire RS-graph to Bob. Their lower bound construction breaks down, however, in two rounds of communication, which reflects the two-pass streaming setting. In this case, the selector edges can easily be learnt in the first pass, which reveals the identity of the important induced matching. In the second pass, the edges of the important induced matching can then be collected. New ideas are thus needed for better hiding the important edges in the two-pass/two-round setting.

Assadi's two-pass lower bound overcomes this hurdle by replacing Bob's selector edges with alternating paths in a clever way using XOR gadgets that are themselves based on RS-graphs as they have been recently used in works for proving lower bounds for exact MBM in the multi-pass streaming setting [AR20, CKP+21, AN21] (see also [AB21]). The construction is such that Alice's important induced matching can only be identified if essentially all edges of the XOR gadgets are stored in the first pass. If, however, less space is allowed, the identity of the important induced matching is still unclear at the beginning of the second pass, and, therefore, learning even only a constant fraction of the edges of the important induced matching in the second pass practically requires storing Alice's entire RS-graph in the second pass.

In our construction, we give up on the property that Alice only holds a single RS-graph. Instead, Alice holds a large number $t_g \cdot r_g$ of edge-disjoint RS-graphs that can be partitioned into t_g groups, each of size r_g, such that the RS-graphs in each group are vertex-disjoint. Our construction is best regarded as a three-party communication game. Alice holds the RS-graphs, Bob holds selector edges that render one induced matching important for each RS-graph, and Charlie holds selector edges that render one group of RS-graphs important. The challenge stems from the fact that, in the first pass, without knowledge of Charlie's selector edges, learning even only a constant fraction of Bob's selector edges from the important group of RS-graphs requires storing practically all of Bob's selector edges, which requires too much

space. Hence, only Charlie's selector edges, and thus the important group of RS-graphs are learnt by the end of the first pass. It follows then that, in the second pass, the identity of the important induced matching for each RS-graph in the important group is still not known, and essentially all edges of Alice's RS-graphs in the important group need to be stored to identify even just a linear fraction of the important edges.

6.1.4 Further Related Work

Recently, Assadi and Sundaresan [AS23a] extended Assadi's two-pass lower bound [Ass22] to multiple passes. Assuming that Ruzsa-Szemerédi graphs with $n^{\Omega(1)}$ induced matchings of linear sizes exist, Assadi and Sundaresan show that semi-streaming algorithms for computing a $(1-\epsilon)$-approximation to MM require $\Omega(\log(\frac{1}{\epsilon}))$ passes, for small $\epsilon > 0$.

Furthermore, various multi-pass lower bounds for exact MM are known. Guruswami and Onak [GO16] showed that space $n^{1+\Omega(\frac{1}{p})}/p^{O(1)}$ is needed to compute a MM in p passes. Assadi and Raz [AR20] subsequently showed that space $\Omega(n^{2-o(1)})$ is required if two passes are allowed. This was then extended by Chen et al. [CKP+21] to hold even if the algorithm is allowed $o(\sqrt{\log n})$ passes.

6.1.5 Outline

In Section 6.2, we give notation, introduce the communication models used in this work, and give useful facts involving mutual information. We present the HiddenStrings problem in Section 6.3 and determine its two-round communication complexity. Then, in Section 6.4 we establish our main lower bound result. Finally, we conclude in Section 6.5 with open problems.

6.2 Preliminaries

Throughout this work, log denotes the binary logarithm. For any positive integer t, we define $[t] := \{1, 2, \ldots, t\}$. For any t-bit-string $Y \in \{0,1\}^t$ and index $J \in [t]$, the Jth bit of Y is denoted as $Y[J]$.

Information Theory and Important Facts

Let A, B, C, D be jointly distributed random variables according to a distribution $\mathcal{D}$. We denote the mutual information of A and B by $\mathcal{I}_{\mathcal{D}}(A;B)$, and the conditional mutual information of A and B conditioned on C by $\mathcal{I}_{\mathcal{D}}(A;B \mid C)$. The Shannon Entropy of A is denoted by $H_{\mathcal{D}}(A)$. We omit the subscript $\mathcal{D}$ when the distribution is clear from context.

In this work, we will use the following important facts:

1. (Information Expectation) $\mathcal{I}(A;B \mid C,D) = \mathbb{E}_D\left[\mathcal{I}(A;B \mid C, D=d)\right]$ [CT06, Chapter 2.5]
2. (Independent Conditioning) Let A and D be independent conditioned on C. Then $\mathcal{I}(A;B \mid C) \leq \mathcal{I}(A;B \mid C,D)$ (e.g., see [AKL16, Claim 2.2] for a proof)
3. (Information Chain Rule) $\mathcal{I}(A;B,C \mid D) = \mathcal{I}(A;B \mid C,D) + \mathcal{I}(A;C \mid D)$ [CT06, Theorem 2.5.2]
4. (Data Processing Inequality) Let A, B and C be a Markov chain such that $A \to B \to C$. Then, $\mathcal{I}(A;C) \leq \mathcal{I}(A;B)$ [CT06, Theorem 2.8.1]

Communication Complexity and Information Cost

We consider the two-party randomised communication model as defined by Yao [Yao79]. Let $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ be sets and $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a function. The parties Alice and Bob are each given inputs $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$, respectively, and their goal is to compute $f(X,Y)$ by exchanging messages according to a randomised protocol π using public (shared) and private randomness. Let $\pi_{\text{out}}(X,Y)$ be the output of the protocol. π is called a δ-error protocol for function f if $\Pr[\pi_{\text{out}}(X,Y) = f(X,Y)] \geq 1-\delta$ for all input pairs $(X,Y) \in \mathcal{X} \times \mathcal{Y}$.

The transcript of the messages exchanged in protocol π is ambiguously also denoted by π. The *communication cost* of π is defined as the maximum transcript length over all possible inputs and executions of the protocol. Then, the δ-error *communication complexity* [Yao79] of f is the minimum communication cost over all δ-error protocols for f.

The *information cost* [CSWY01, BJKS02] of π is the amount of information revealed about the inputs, over some distribution of the random variables, by the messages communicated. Relevant to our work is the notion of (internal) information cost with respect to Alice's or Bob's input: We define Alice's information cost of protocol π as $\mathsf{ICost}^A_{\mathcal{D}}(\pi) = \mathcal{I}_{\mathcal{D}}(X : \pi \mid Y, R)$, and, similarly, Bob's information cost of π as $\mathsf{ICost}^B_{\mathcal{D}}(\pi) = \mathcal{I}_{\mathcal{D}}(Y : \pi \mid X, R)$, where $\mathcal{D}$ is a distribution of the random variables and R is the public randomness used by π.

6.3 A new communication game: HiddenStrings

In this section, we give a *one-way three-party* communication game called HiddenStrings and determine its *two-round* communication complexity.

Let Alice, Bob and Charlie be the three parties and let t_g, r_g, t_ℓ, r_ℓ be positive integers. Alice holds the bit strings $X[i,j,k] \in \{0,1\}^{r_\ell}$, for every $i \in [t_g], j \in [r_g]$ and $k \in [t_\ell]$, Bob holds indices $K[i,j] \in [t_\ell]$, for every $i \in [t_g]$ and $j \in [r_g]$, and Charlie holds $I^* \in [t_g]$. The goal is for second-round Charlie to output the bit strings $X[I^*, j, K[I^*, j]]$, for every $j \in [r_g]$.

As previously discussed, the two protocols that either communicate only Charlie's input

or both Bob and Charlie's inputs in the first round yield communication costs of $\tilde{O}(r_g \cdot t_\ell \cdot r_\ell)$ and $\tilde{O}(t_g \cdot r_g + r_g \cdot r_\ell)$, respectively. As our main result of this section, we prove that these protocols are best possible:

Theorem 6.3 (cf. Theorem 7). *For any constant $\delta < \frac{1}{2}$, any δ-error two-round protocol for* HiddenStrings *requires $\Omega(\min\{t_g \cdot r_g, r_g \cdot t_\ell \cdot r_\ell\})$ bits of communication.*

We will see that the uniform input distribution $\mathcal{D}_{\mathsf{HS}}$ as depicted in Figure 6.2 is a hard distribution for HiddenStrings and is used for proving Theorem 6.3.

Distribution $\mathcal{D}_{\mathsf{HS}}$:

1. (Alice's Input) For each $i \in [t_g]$, $j \in [r_g]$, and $k \in [t_\ell]$, uniformly and independently sample $X[i, j, k]$ from the set $\{0, 1\}^{r_\ell}$.
2. (Bob's Input) For each $i \in [t_g]$ and $j \in [r_g]$, uniformly and independently sample $K[i, j]$ from the set $[t_\ell]$.
3. (Charlie's Input) Uniformly and independently sample I^* from the set $[t_g]$.

Figure 6.2: Uniform input distribution $\mathcal{D}_{\mathsf{HS}}$ for HiddenStrings.

To prove our main result, we show in Subsection 6.3.1 that any protocol π_{HS} for HiddenStrings can be used to construct a protocol π_{I} for Index. Then, in Subsection 6.3.2, using the information cost trade-off result for Index (Proposition 6.2), we prove that any protocol for HiddenStrings either reveals $\Omega(t_g \cdot r_g)$ bits about K or $\Omega(r_g \cdot r_\ell \cdot t_\ell)$ bits about X, which proves our result.

6.3.1 Protocol π_{I} for Index_{t_ℓ}

For any positive constant $\delta < \frac{1}{2}$, let π_{HS} be a δ-error two-round protocol for HiddenStrings over $\mathcal{D}_{\mathsf{HS}}$. We show in Figure 6.3 that, using protocol π_{HS}, we can construct a δ-error two-round protocol π_{I} for Index_{t_ℓ} on the uniform distribution of inputs $\mathcal{D}_{\mathsf{I}}$. By construction of the protocol π_{I}, it is immediate that π_{I} succeeds whenever π_{HS} succeeds. The protocol π_{I} thus has a success probability of at least $1 - \delta$.

Let $\pi_{\mathsf{HS}} = \{\pi_{\mathsf{HS}}^{A1}, \pi_{\mathsf{HS}}^{B1}, \pi_{\mathsf{HS}}^{C1}, \pi_{\mathsf{HS}}^{A2}, \pi_{\mathsf{HS}}^{B2}\}$ denote the transcript of messages sent by the three parties Alice, Bob, and Charlie in protocol π_{HS} with public randomness R_{HS}, and let $\pi_{\mathsf{I}} = \{\pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}, \pi_{\mathsf{I}}^{A2}\}$ denote the transcript of the messages sent by the two parties Alice and Bob in the constructed protocol π_{I} with public randomness R_{I}. Then, $R_{\mathsf{I}} = \{I^*, J^*, \varphi^*, R_{\mathsf{HS}}\}$, $\pi_{\mathsf{I}}^{A1} = \pi_{\mathsf{HS}}^{A1}$, $\pi_{\mathsf{I}}^{B1} = \pi_{\mathsf{HS}}^{B1}$, and $\pi_{\mathsf{I}}^{A2} = \pi_{\mathsf{HS}}^{A2}$, and, in particular, the messages π_{HS}^{C1} and π_{HS}^{B2} are not explicitly sent in π_{I} since, in π_{I}, Alice simulates first-round Charlie of π_{HS} and Bob simulates second-round Charlie of π_{HS}.

Input: $(Y, J) \sim \mathcal{D}_{\mathsf{I}}$ and protocol π_{HS}.

Protocol π_{I}:

1. Using public randomness, Alice and Bob uniformly sample $I^* \in [t_g]$, $J^* \in [r_g]$, and $\varphi^* \in [r_\ell]$.
2. Using private randomness, for each $i \in [t_g]$, $j \in [r_g]$, and $k \in [t_\ell]$, Alice uniformly samples $X[i, j, k] \in \{0, 1\}^{r_\ell}$.
3. Using private randomness, for each $i \in [t_g]$ and $j \in [r_g]$, Bob uniformly samples $K[i, j] \in [t_\ell]$.
4. For each $k \in [t_\ell]$, Alice replaces $X[I^*, J^*, k][\varphi^*]$ with $Y[k]$. Bob replaces $K[I^*, J^*]$ with J.
5. Alice and Bob run π_{HS} in two rounds where, in the second round, Alice simulates first-round Charlie in π_{HS} and Bob simulates second-round Charlie in π_{HS}. This is possible since both parties know I^*.
6. The output of π_{HS} is $X[I^*, j, K[I^*, j]]$, for every $j \in [r_g]$. Bob then outputs $X[I^*, J^*, K[I^*, J^*]][\varphi^*]$ as the predictor of $Y[J]$.

Figure 6.3: A two-round protocol π_{I} for Index_{t_ℓ} using π_{HS}.

6.3.2 Analysis: Invoking the Information Cost Trade-off of Index

Let s be the maximum number of bits communicated by π_{HS} in the first round. We show that, when $s = O(t_g \cdot r_g)$, Alice's second round message must communicate $\Omega(r_g \cdot r_\ell \cdot t_\ell)$ bits, which implies Theorem 6.3.

To this end, using fundamental facts from Information Theory (see Section 6.2), we argue bounds on the information cost of π_{I}, first w.r.t. Bob then w.r.t. Alice. This allows us to invoke the information cost trade-off result for Index (Proposition 6.2) and complete the proof of Theorem 6.3.

Lemma 6.5. *For protocol π_{I} described in Figure 6.3, the following holds:*

$$\mathsf{ICost}^{B}_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) = \mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(J; \pi_{\mathsf{I}} \mid Y, R_{\mathsf{I}}) \leq \frac{s}{t_g \cdot r_g} \ ,$$

where s is the maximum number of bits communicated by π_{HS} in the first round.

Proof. Since $R_{\mathsf{I}} = \{I^*, J^*, \varphi^*, R_{\mathsf{HS}}\}$, $\pi_{\mathsf{I}}^{A1} = \pi_{\mathsf{HS}}^{A1}$, $\pi_{\mathsf{I}}^{B1} = \pi_{\mathsf{HS}}^{B1}$ and $\pi_{\mathsf{I}}^{A2} = \pi_{\mathsf{HS}}^{A2}$, and, given a uniform random input to Index, the protocol constructs a uniform random input to HiddenStrings, we have that

$$\begin{aligned}
&\mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(J; \pi_{\mathsf{I}} \mid Y, R_{\mathsf{I}}) \\
&\quad = \mathcal{I}_{\mathcal{D}_{\mathsf{HS}}}(K[I^*, J^*]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}, \pi_{\mathsf{I}}^{A2} \mid X[I^*, J^*, 1][\varphi^*], \ldots, X[I^*, J^*, t_\ell][\varphi^*], I^*, J^*, \varphi^*, R_{\mathsf{HS}})
\end{aligned}$$

$$\leq \mathcal{I}_{\mathcal{D}_{\mathsf{HS}}}(K[I^*, J^*]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}, \pi_{\mathsf{I}}^{A2} \mid X, I^*, J^*, \varphi^*, R_{\mathsf{HS}})$$

(by the conditioning fact (Fact 2) since the inputs are mutually independent)

$$= \mathcal{I}_{\mathcal{D}_{\mathsf{HS}}}(K[I^*, J^*]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1} \mid X, I^*, J^*, \varphi^*, R_{\mathsf{HS}}) + \underbrace{\mathcal{I}_{\mathcal{D}_{\mathsf{HS}}}(K[I^*, J^*]; \pi_{\mathsf{I}}^{A2} \mid \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}, X, I^*, J^*, \varphi^*, R_{\mathsf{HS}})}_{=0}$$

(by the chain rule (Fact 3))

$$= \mathcal{I}_{\mathcal{D}_{\mathsf{HS}}}(K[I^*, J^*]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1} \mid X, I^*, J^*, \varphi^*, R_{\mathsf{HS}}) \ .$$

(by the data processing inequality (Fact 4) since π_{I}^{A2} is a function of π_{I}^{B1}, X, I^*, and R_{HS})

It now follows that, since I^* and J^* are each entirely independent of $\pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}, X, K, \varphi^*, R_{\mathsf{HS}}$ and each other, and since the inputs are mutually independent with regards to the conditioning in the mutual information term, we can use the expectation (Fact 1) and conditioning (Fact 2) facts followed by the chain rule (Fact 3) – first w.r.t. J^* then w.r.t. I^* – to show that

$$\mathcal{I}(K[I^*, J^*]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1} \mid X, I^*, J^*, \varphi^*, R_{\mathsf{HS}})$$

$$= \frac{1}{r_g} \cdot \sum_{j^*=1}^{r_g} \mathcal{I}(K[I^*, j^*]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1} \mid X, I^*, J^* = j^*, \varphi^*, R_{\mathsf{HS}})$$

(by Fact 1 since J^* is uniformly distributed)

$$= \frac{1}{r_g} \cdot \sum_{j^*=1}^{r_g} \mathcal{I}(K[I^*, j^*]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1} \mid X, I^*, \varphi^*, R_{\mathsf{HS}})$$ (by the independence of J^*)

$$\leq \frac{1}{r_g} \cdot \sum_{j^*=1}^{r_g} \mathcal{I}(K[I^*, j^*]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1} \mid X, I^*, K[I^*, 1], \ldots K[I^*, j^* - 1], \varphi^*, R_{\mathsf{HS}})$$

(by Fact 2 since the inputs are mutually independent)

$$= \frac{1}{r_g} \cdot \mathcal{I}(K[I^*, 1], \ldots, K[I^*, r_g]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1} \mid X, I^*, \varphi^*, R_{\mathsf{HS}})$$ (by Fact 3)

$$\leq \frac{1}{t_g \cdot r_g} \cdot \mathcal{I}(K; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1} \mid X, \varphi^*, R_{\mathsf{HS}})$$

(by a similar string of arguments using Facts 1, 2 and 3 w.r.t. I^*)

$$\leq \frac{1}{t_g \cdot r_g} \cdot H(\pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1})$$ (since conditioning can only reduce the entropy)

$$\leq \frac{s}{t_g \cdot r_g} \ ,$$

where the last inequality holds because $\pi_{\mathsf{I}}^{A1} = \pi_{\mathsf{HS}}^{A1}$ and $\pi_{\mathsf{I}}^{B1} = \pi_{\mathsf{HS}}^{B1}$ and the total first-round communication in π_{HS} is at most s bits. □

Lemma 6.6. *For protocol π_{I} described in Figure 6.3, the following holds:*

$$\mathsf{ICost}_{\mathcal{D}_{\mathsf{I}}}^{A}(\pi_{\mathsf{I}}) = \mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(Y; \pi_{\mathsf{I}} \mid J, R_{\mathsf{I}}) \leq \frac{s}{t_g \cdot r_g \cdot r_\ell} + \frac{1}{r_g \cdot r_\ell} \cdot H(\pi_{\mathsf{HS}}^{A2}) \ ,$$

where s is the maximum number of bits communicated by π_{HS} in the first round.

Proof. Since $R_{\mathsf{I}} = \{I^*, J^*, \varphi^*, R_{\mathsf{HS}}\}$, $\pi_{\mathsf{I}}^{A1} = \pi_{\mathsf{HS}}^{A1}$, $\pi_{\mathsf{I}}^{B1} = \pi_{\mathsf{HS}}^{B1}$ and $\pi_{\mathsf{I}}^{A2} = \pi_{\mathsf{HS}}^{A2}$, and, given a uniform random input to Index, the protocol constructs a uniform random input to HiddenStrings, we have that

$$\begin{aligned}
&\mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(Y; \pi_{\mathsf{I}} \mid J, R_{\mathsf{I}}) \\
&\quad = \mathcal{I}_{\mathcal{D}_{\mathsf{HS}}}(X[I^*, J^*, 1][\varphi^*], \ldots, X[I^*, J^*, t_\ell][\varphi^*]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}, \pi_{\mathsf{I}}^{A2} \mid K[I^*, J^*], I^*, J^*, \varphi^*, R_{\mathsf{HS}}) \\
&\quad \leq \mathcal{I}_{\mathcal{D}_{\mathsf{HS}}}(X[I^*, J^*, 1][\varphi^*], \ldots, X[I^*, J^*, t_\ell][\varphi^*]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}, \pi_{\mathsf{I}}^{A2} \mid K, I^*, J^*, \varphi^*, R_{\mathsf{HS}}) \ . \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(by Fact 2 since the inputs are mutually independent)}
\end{aligned}$$

It now follows that, since J^* and φ^* are each entirely independent of $\pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}, \pi_{\mathsf{I}}^{A2}, X, K, I^*, R_{\mathsf{HS}}$ and each other, and since the inputs are mutually independent with regards to the conditioning in the mutual information term, we can follow the steps in the proof of Lemma 6.5 and use the expectation (Fact 1) and conditioning (Fact 2) facts followed by the chain rule (Fact 3) – first w.r.t. φ^* then w.r.t. J^* – to show that

$$\begin{aligned}
&\mathcal{I}_{\mathcal{D}_{\mathsf{HS}}}(X[I^*, J^*, 1][\varphi^*], \ldots, X[I^*, J^*, t_\ell][\varphi^*]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}, \pi_{\mathsf{I}}^{A2} \mid K, I^*, J^*, \varphi^*, R_{\mathsf{HS}}) \\
&\quad \leq \frac{1}{r_\ell} \cdot \mathcal{I}(X[I^*, J^*, 1], \ldots, X[I^*, J^*, t_\ell]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}, \pi_{\mathsf{I}}^{A2} \mid K, I^*, J^*, R_{\mathsf{HS}}) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(by Facts 1, 2 and 3 w.r.t. } \varphi^*) \\
&\quad \leq \frac{1}{r_g \cdot r_\ell} \cdot \mathcal{I}(X[I^*, 1, 1], \ldots, X[I^*, r_g, t_\ell]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}, \pi_{\mathsf{I}}^{A2} \mid K, I^*, R_{\mathsf{HS}}) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(by Facts 1, 2 and 3 w.r.t. } J^*) \\
&\quad \leq \frac{1}{r_g \cdot r_\ell} \cdot \Big(\mathcal{I}(X[I^*, 1, 1], \ldots, X[I^*, r_g, t_\ell]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1} \mid K, I^*, R_{\mathsf{HS}}) \qquad \text{(by Fact 3)} \\
&\qquad\qquad + \mathcal{I}(X[I^*, 1, 1], \ldots, X[I^*, r_g, t_\ell]; \pi_{\mathsf{I}}^{A2} \mid \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}, K, I^*, R_{\mathsf{HS}}) \Big) \\
&\quad \leq \frac{1}{r_g \cdot r_\ell} \cdot \Big(\mathcal{I}(X[I^*, 1, 1], \ldots, X[I^*, r_g, t_\ell]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1} \mid K, I^*, R_{\mathsf{HS}}) + H(\pi_{\mathsf{HS}}^{A2}) \Big) \ . \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(since } \pi_{\mathsf{I}}^{A2} = \pi_{\mathsf{HS}}^{A2})
\end{aligned}$$

We complete the proof by bounding the remaining mutual information term. Since I^* is entirely independent of $\pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}, X, K, R_{\mathsf{HS}}$, and since the inputs are mutually independent with regards to the conditioning in the mutual information term, we have that

$$\begin{aligned}
&\mathcal{I}(X[I^*, 1, 1], \ldots, X[I^*, r_g, t_\ell]; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1} \mid K, I^*, R_{\mathsf{HS}}) \\
&\quad \leq \frac{1}{t_g} \cdot \mathcal{I}(X; \pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1} \mid K, R_{\mathsf{HS}}) \qquad\qquad \text{(by Facts 1, 2 and 3 w.r.t. } I^*) \\
&\quad \leq \frac{1}{t_g} \cdot H(\pi_{\mathsf{I}}^{A1}, \pi_{\mathsf{I}}^{B1}) \leq \frac{s}{t_g} \ ,
\end{aligned}$$

where the last inequality holds because $\pi_{\mathsf{I}}^{A1} = \pi_{\mathsf{HS}}^{A1}$ and $\pi_{\mathsf{I}}^{B1} = \pi_{\mathsf{HS}}^{B1}$ and the total first-round communication in π_{HS} is at most s bits. □

The proof of Theorem 6.3 now follows from Lemmas 6.5 and 6.6 and the Information cost trade-off of Index (Proposition 6.2).

Proof of Theorem 6.3. Suppose that the maximum number of bits communicated by π_{HS} in the first round is $s = O(t_g \cdot r_g)$. By Lemma 6.5, we have that:

$$\mathsf{ICost}^{B}_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) \leq \frac{s}{t_g \cdot r_g} = O(1) \ .$$

Then, applying the information cost trade-off for Index (Proposition 6.2) to the previous inequality, we obtain

$$\mathsf{ICost}^{A}_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) = \Omega(t_\ell) \ .$$

Combining the previous equality with the statement of Lemma 6.6, and using the assumption that $s = O(t_g \cdot r_g)$, we obtain:

$$\begin{aligned}\Omega(t_\ell) = \mathsf{ICost}^{A}_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) &\leq \frac{s}{t_g \cdot r_g \cdot r_\ell} + \frac{1}{r_g \cdot r_\ell} \cdot H(\pi^{A2}_{\mathsf{HS}}) \\ &= O(\frac{1}{r_\ell}) + \frac{1}{r_g \cdot r_\ell} \cdot H(\pi^{A2}_{\mathsf{HS}}) \ ,\end{aligned}$$

which further implies $H(\pi^{A2}_{\mathsf{HS}}) = \Omega(r_g \cdot r_\ell \cdot t_\ell)$. Hence, we have proved that either $s = \Omega(t_g \cdot r_g)$ or $H(\pi^{A2}_{\mathsf{HS}}) = \Omega(r_g \cdot r_\ell \cdot t_\ell)$ holds, which implies the result. □

6.4 Two-Pass Maximum Matching Lower Bound

In this section, we prove our main lower bound result:

Theorem 6.1 (cf. Theorem 6). For any constant $\epsilon > 0$, any constant-error two-pass $(\frac{8}{9} + \epsilon)$-approximation streaming algorithm for MBM requires $n^{1+\Omega(\frac{1}{(\log\log n)^2})}$ bits of space, where n is the number of vertices of the input graph.

Our result is achieved by showing that any two-pass streaming algorithm for MBM with approximation factor $\frac{8}{9} + \epsilon$ can be used to obtain a protocol that solves HiddenStrings. In Subsection 6.4.1, we will first show how a protocol for HiddenStrings is obtained from a two-pass algorithm for MBM, and then give the analysis in Subsection 6.4.2.

6.4.1 From MBM to HiddenStrings

For the remainder of this section, let $\mathcal{A}$ be a two-pass streaming algorithm for MBM with approximation factor $\frac{8}{9} + \epsilon$, for any constant $\epsilon > 0$. Furthermore, fix parameters t_g, r_g, t_ℓ, r_ℓ and let (X, K, I^*) be a HiddenStrings instance, i.e., let $X[i,j,k] \in \{0,1\}^{r_\ell}$, for every $i \in [t_g], j \in [r_g], k \in [t_\ell]$, let $K[i,j] \in [t_\ell]$, for every $i \in [t_g], j \in [r_g]$, and let $I^* \in [t_g]$. Based on their inputs

(X, K, I^*), Alice, Bob, and Charlie construct multiple graphs on which they simulate parallel runs of algorithm $\mathcal{A}$.

The construction of these graphs make use of a Rusza-Szemerédi graph construction originally given by [FLN+02] and subsequently improved by [GKK12]. We will instantiate these RS-graphs on very specific sizes and therefore give a proof that graphs of the required sizes indeed exist:

Proposition 6.7. *For any small constant $\gamma > 0$ and for every large enough n', there exists a $(2n)$-vertex balanced bipartite (r, t)-RS-graph where $r = (\frac{1}{2} - \gamma) \cdot n$ and $t = n^{\Omega(\frac{1}{\log \log n})}$, such that $n \leq n' < n \log^2 n'$.*

Proof. We use the construction given in [GKK12]. Their construction is such that the bipartitions of the resulting RS-graph consist of m^{2m} vertices each, for integers m. We will argue that, for every large enough n', we can find a suitable m such that $n := m^{2m}$ is not too far from n'.

Let m be such that

$$m^{2m} \leq n' < (m+1)^{2(m+1)} ,$$

and let $n = m^{2m}$. Then, $n \leq n'$ and it remains to prove that $n' < n \log^2(n')$. To this end, we compute:

$$\frac{n'}{n} < \frac{(m+1)^{2m+2}}{m^{2m}} = (m+1)^2 \cdot \left(\frac{m+1}{m}\right)^{2m} \leq (m+1)^2 \cdot e^{\frac{1}{m} \cdot 2m} = (m+1)^2 e^2 .$$

Last, observe that

$$\log^2(n') \geq \log^2(n) = \log^2(m^{2m}) = (2m \cdot \log(m))^2 \geq (m+1)^2 e^2 ,$$

which completes the proof. □

Each graph $G := G(Y, \sigma^{r_g}, \sigma^{t_\ell}, \sigma^{r_\ell}, X, K, I^*)$ constructed by Alice, Bob, and Charlie is parameterized by:

1. Uniform random bits $Y[i, j, k][\varphi] \in \{0, 1\}$, for every $i \in [t_g], j \in [r_g], k \in [t_\ell]$, and $\varphi \in [r_\ell]$;
2. Three uniform random permutations $\sigma^{r_g} : [r_g] \to [r_g]$, $\sigma^{t_\ell} : [t_\ell] \to [t_\ell]$, and $\sigma^{r_\ell} : [r_\ell] \to [r_\ell]$;
3. The input (X, K, I^*) to HiddenStrings.

The bits $X[i, j, k][\varphi]$ will translate into the presence or absence of edges in G. However, instead of directly working with X (and K), we instead work with X' (and K') that we define as follows: First, we let X'' be the bit-string obtained from X by computing the coordinate-wise XOR with the entries in Y. Then, we subsequently permute the entries in X'' and K

with permutations $\sigma^{r_g}, \sigma^{t_\ell}, \sigma^{r_\ell}$ to obtain

$$X'[i, \sigma^{r_g}(j), \sigma^{t_\ell}(k)][\sigma^{r_\ell}(\varphi)] = X''[i,j,k][\varphi] = X[i,j,k][\varphi] \text{ XOR } Y[i,j,k][\varphi], \text{ for every } i,j,k,\varphi \ ,$$
$$K'[i, \sigma^{r_g}(j)] = \sigma^{t_\ell}(K[i,j]), \text{ for every } i,j \ .$$

Hence, we have that $X'[i, \sigma^{r_g}(j), K'[i, \sigma^{r_g}(j)]][\sigma^{r_\ell}(\varphi)] = X''[i,j,K[i,j]][\varphi]$.

Computing the XOR with Y guarantees that the resulting bits X'' are uniform random bits. Applying the permutations has the effect that, whenever algorithm $\mathcal{A}$ reports a matching, its relevant edges may have originated from any of the relevant bits that need to be learnt for solving HiddenStrings.

We further have that, for suitable parameters $n_g \gg n_\ell$ whose values we will determine later, the construction of G is guided by:

1. A (fixed) *local* (r_ℓ, t_ℓ)-RS-graph G_ℓ^{RS} on $2n_\ell$ vertices as in Proposition 6.7, where $r_\ell = (\frac{1}{2} - \gamma) \cdot n_\ell$ and $t_\ell = n_\ell^{1+\Omega(\frac{1}{\log\log n_\ell})}$;
2. A (fixed) *global* (r_g, t_g)-RS-graph G_g^{RS} on vertices $U, V = [n_g]$ as in Proposition 6.7, where $r_g = (\frac{1}{2} - \gamma) \cdot n_g$ and $t_g = n_g^{1+\Omega(\frac{1}{\log\log n_g})}$.

The graph G has bipartitions $A \cup A^*$ and $B \cup B^*$ with $|A| = |B| = n_g \cdot (2n_\ell - r_\ell)$, and $|A^*| = |B^*| = (n_g - r_g) \cdot (2n_\ell - r_\ell)$; hence, it has $n = \Theta(n_g \cdot n_\ell)$ vertices. The vertices A are partitioned into vertex groups $A_u = A_u^1 \cup A_u^2$ with $|A_u^1| = n_\ell$ and $|A_u^2| = n_\ell - r_\ell$ for each $u \in U$, and the vertices B are similarly partitioned into vertex groups $B_v = B_v^1 \cup B_v^2$ with $|B_v^1| = n_\ell$ and $|B_v^2| = n_\ell - r_\ell$ for each $v \in V$.

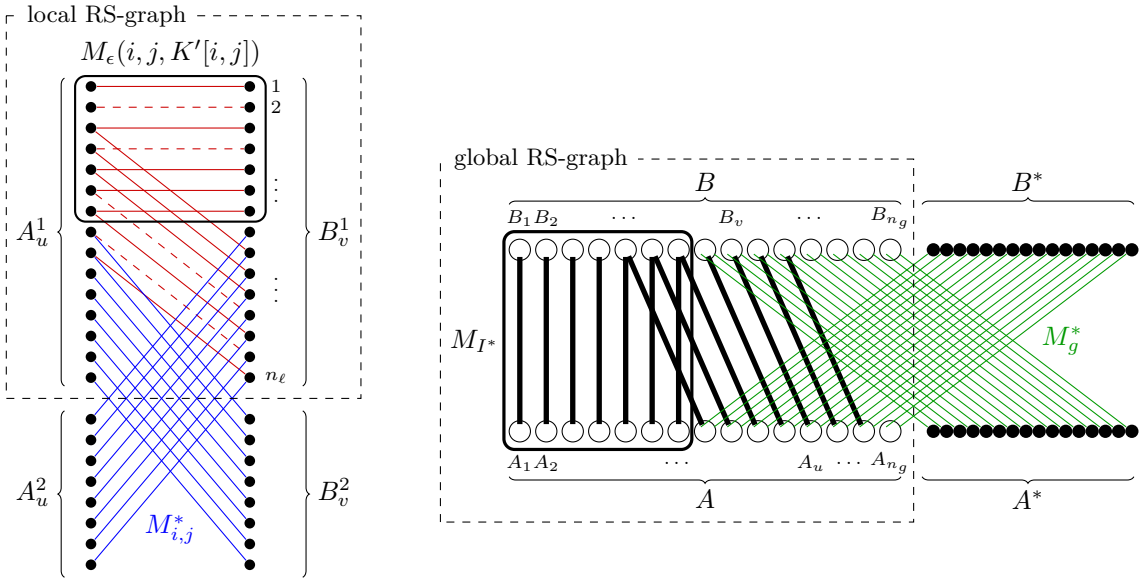
Alice, Bob, and Charlie's edges in G are constructed as follows and are illustrated in Figure 6.4:

Alice's Edges

1. For each $i \in [t_g], j \in [r_g]$, Alice constructs G_ℓ^{RS} w.r.t. vertex sets A_u^1 and B_v^1 where (u,v) is the jth edge of the ith global induced matching in G_g^{RS}. We can now identify every edge e constructed by four parameters i,j,k,φ as the edge that is the φth edge of the kth induced matching of the local RS-graph that corresponds to the jth edge of the ith global induced matching, i.e., $e = (i,j,k,\varphi)$.
2. For each $i \in [t_g], j \in [r_g], k \in [t_\ell], \varphi \in [r_\ell]$:

   If $X'[i,j,k][\varphi] = 0$ **then** delete the edge (i,j,k,φ) with probability 2ϵ.

We will use the following notation: Let $M(i,j,k)$ denote the kth induced matching in the local RS-graph that corresponds to the jth edge of the ith global induced matching *before* Alice's deletions, and let $M_\epsilon(i,j,k)$ denote the corresponding matching *after* Alice's deletions.



(a) A local structure of G. The (red) edges of G_ℓ^{RS} are constructed by Alice where the dashed ones represent the randomly deleted edges. A thick (black) outline highlights a special local induced matching $M(i,j,K'[i,j])$. The remaining (blue) edges represent the matching $M^*_{i,j}$ constructed by Bob.

(b) The global structure of G. The unfilled circular nodes represent the vertex groups that partition A and B, each of which represent a vertex in G_g^{RS}. The thick (black) edges between the vertex groups, which exist for every edge $(u,v) \in G_g^{\text{RS}}$, represent pairs of vertex groups (A_u, B_v) that have local edges between them – these edges are illustrated in Figure 6.4a. A thick (black) outline highlights the special global induced matching M_{I^*}. The remaining (green) edges represent the matching M^*_g constructed by Charlie.

Figure 6.4: An illustration of the graph $G = G(Y, \sigma^{r_g}, \sigma^{t_\ell}, \sigma^{r_g}, X, K, I^*)$. It is made up of local structures, shown in Figure 6.4a, and a global structure, shown in Figure 6.4b. Each are guided by local RS-graph G_ℓ^{RS} and a global RS-graph G_g^{RS}, respectively, to construct the edges of Alice (in red), Bob (in blue) and Charlie (in green). In both cases, the RS-graph is indicated by a dashed outline where its edges are illustrated w.r.t. X' and K', i.e., after the XOR operation and the respective permutations σ^{r_g}, σ^{t_ℓ} and σ^{r_ℓ}. In both illustrations, the (black) filled circular nodes represent vertices in G.

Bob's Edges

For each $i \in [t_g], j \in [r_g]$, Bob constructs a perfect matching $M^*_{i,j}$ to be the union of two perfect matchings, one between A_u^2 and $B_v^1 \backslash B(M(i,j,K'[i,j]))$, and one between $A_u^1 \backslash A(M(i,j,K'[i,j]))$ and B_v^2 where (u,v) is the jth edge of the ith global induced matching in G_g^{RS}.

Charlie's Edges

Let M_{I^*} be the I^*th induced matching in the global RS-graph. Let $U' = U \setminus U(M_{I^*})$ and let $V' = V \setminus V(M_{I^*})$. Charlie adds the edges M^*_g to be the union of two perfect matchings, one between $A' = \cup_{u \in U'} A_u$ and B^*, and one between A^* and $B' = \cup_{v \in V'} B_v$.

Reduction

In order to obtain a communication protocol π_{HS} for HiddenStrings from $\mathcal{A}$ we proceed as follows: Alice, Bob, and Charlie run the algorithm $\mathcal{A}$ $\Theta(\frac{1}{\epsilon} \cdot \log^2(n_g \cdot n_\ell))$ times in parallel by forwarding the memory states of the parallel runs of $\mathcal{A}$ to the next party. In each of these runs, a new set of uniform random permutations are chosen (using public randomness) to construct the input graph G. However, across all runs, the same random bits Y (using public randomness) are used. In addition to the memory states, I^* is forwarded from Charlie in the first round to Bob in the second round, who then uses this to forward $K[I^*, j]$, for all j, to the final party Charlie. We will see that Charlie requires knowledge of these bits to create the output to HiddenStrings. The protocol π_{HS} thus has a communication cost of $O(|\mathcal{A}| \cdot \frac{1}{\epsilon} \log^2(n_g \cdot n_\ell) + \log t_g + r_g \cdot \log t_\ell)$, where $|\mathcal{A}|$ is the space used by $\mathcal{A}$, and the lower bound in the previous section yields a lower bound on $|\mathcal{A}|$.

6.4.2 Analysis

We will first prove that, with high probability, the graph G contains a matching of large size (Lemma 6.9). Then, we will argue that the algorithm $\mathcal{A}$ is guaranteed to output at least $\epsilon \cdot n_g \cdot n_\ell$ edges from the matchings $M_\epsilon(I^*, j, K'[I^*, j])$, for all j (Lemma 6.10), which we refer to as the *special matchings*. Recall that the random permutations $\sigma^{r_g}, \sigma^{t_\ell}, \sigma^{r_\ell}$ guarantee that each special matching edge may originate from any of the relevant bits that need to be learnt to solve HiddenStrings. In particular, by fixing $\epsilon \cdot n_g \cdot n_\ell$ special matching edges from the output of $\mathcal{A}$, we show that a relevant bit, i.e., $X''[I^*, j, K[I^*, j]][\varphi]$, for any j and φ, has probability $p_1 = \Theta(\epsilon)$ (over the choices of the random permutations) to be reflected as a special matching edge when its value is a 1 and probability $p_0 = (1 - 2\epsilon) \cdot p_1$ when its value is a 0 (Lemma 6.11). Therefore, by running $\mathcal{A}$ in parallel $O(\frac{1}{\epsilon} \cdot \log^2(n_g \cdot n_\ell))$ times, we can distinguish the cases for all relevant bits and prove our main result (Theorem 6.1).

Lemma 6.8. *The joint size of the special matchings is*

$$\Big| \bigcup_{j \in [r_g]} M_\epsilon(I^*, j, K'[I^*, j]) \Big| = (1 - \epsilon) \cdot r_g \cdot r_\ell \pm o(r_g \cdot r_\ell) \ ,$$

with probability at least $1 - \frac{1}{\text{poly}(r_g \cdot r_\ell)}$.

Proof. Observe that, for any i, j, k, the φth edge in $M(i, j, k)$ corresponds to the bit $X'[i, j, k][\varphi]$ in that, if $X'[i, j, k][\varphi] = 0$ then the edge is deleted with probability 2ϵ. Since $X'[i, j, k][\varphi]$ is a uniform random binary variable, the edge is deleted with probability ϵ. Hence, in expectation, $\epsilon \cdot r_g \cdot r_\ell$ edges of the special matchings are deleted and the result then follows from concentration bounds. □

Lemma 6.9. *With probability at least* $1 - \frac{1}{\text{poly}(n_g \cdot n_\ell)}$*, the size of a maximum matching in* G *is at least*

$$\left(\frac{9}{4} - \frac{\epsilon}{4}\right) \cdot n_g \cdot n_\ell \ .$$

Proof. To prove the statement, we construct a maximum matching M^* of the claimed size. To this end, we use a folklore algorithm for finding a maximum matching in a graph, which repeats the following procedure: If there exists an edge uv such that at least one of its endpoints has degree 1 then add the edge to the matching and remove u and v and their incident edges from the graph. If there is no such edge left then compute a maximum matching in the remaining graph by different means. We will see that the graph G is such that there is always an edge with one endpoint of degree 1 left until the graph becomes empty.

First, observe that all edges of Charlie's matching M_g^* have an endpoint of degree 1 (the endpoints in $A^* \cup B^*$). We thus add those edges to the maximum matching M^*. Then, the remaining graph $G \backslash V(M_g^*)$ consists only of the local RS-graphs that correspond to the I^*th global induced matching. Therefore, Bob's remaining edges $\bigcup_{j \in [r_g]} M^*_{I^*,j}$ now form a matching where all edges have an endpoint of degree 1 and we add these to the maximum matching M^*. The remaining graph $G \backslash V(M_g^* \cup (\bigcup_{j \in [r_g]} M^*_{I^*,j}))$ consists only of the edges of the special matchings $M_\epsilon(I^*, j, K'[I^*, j])$, for all j, which we add to M^* as well. Therefore, the size of M^* is as follows:

$$\begin{aligned}
|M^*| &= |M_g^*| + |\bigcup_{j \in [r_g]} M^*_{I^*,j}| + |\bigcup_{j \in [r_g]} M_\epsilon(I^*, j, K'[I^*, j])| \\
&= 2 \cdot (n_g - r_g) \cdot (2n_\ell - r_\ell) + r_g \cdot 2 \cdot (n_\ell - r_\ell) + |\bigcup_{j \in [r_g]} M_\epsilon(I^*, j, K'[I^*, j])| \\
&\qquad\qquad \text{(by construction)} \\
&\geq 2 \cdot (n_g - r_g) \cdot (2n_\ell - r_\ell) + r_g \cdot 2 \cdot (n_\ell - r_\ell) + (1 - \epsilon) \cdot r_g \cdot r_\ell - o(r_g \cdot r_\ell) \\
&\qquad\qquad \text{(by Lemma 6.8)} \\
&= 2 \cdot (\frac{1}{2} + \gamma) \cdot (\frac{3}{2} + \gamma) \cdot n_g \cdot n_\ell + 2 \cdot (\frac{1}{2} - \gamma) \cdot (\frac{1}{2} + \gamma) \cdot n_g \cdot n_\ell \qquad \text{(by Proposition 6.7)} \\
&\quad + (1 - \epsilon) \cdot (\frac{1}{2} - \gamma)^2 \cdot n_g \cdot n_\ell - o(n_g \cdot n_\ell) \\
&= (2 + 4\gamma + (1 - \epsilon) \cdot (\frac{1}{4} - \gamma + \gamma^2)) \cdot n_g \cdot n_\ell - o(n_g \cdot n_\ell) \\
&\geq (\frac{9}{4} - \frac{\epsilon}{4}) \cdot n_g \cdot n_\ell
\end{aligned}$$

with probability at least $1 - \frac{1}{\text{poly}(r_g \cdot r_\ell)}$. □

Lemma 6.10. *Suppose that* $\epsilon \leq 1$ *and* $\gamma \leq \frac{1}{8}\epsilon$*. Then, with probability at least* $1 - \frac{1}{\text{poly}(n_g \cdot n_\ell)}$*, any* $(\frac{8}{9} + \epsilon)$*-approximate matching* M *in* G *contains at least* $\epsilon \cdot n_g \cdot n_\ell$ *edges from the special matchings* $M_\epsilon(I^*, j, K'[I^*, j])$ *for all* $j \in [r_g]$*.*

Proof. Let M be any matching in G and let $M_s = M \cap \left(\bigcup_{j\in[r_g]} M(I^*, j, K'[I^*, j])\right)$ be the special matching edges that are also in M. Let $G_{\text{res}} := G \setminus \left(\bigcup_{j\in[r_g]} M(I^*, j, K'[I^*, j])\right)$ be the graph obtained from G when removing all special edges. Then,

$$|M| \leq |M_s| + \mu(G_{\text{res}}) \ , \tag{6.1}$$

where $\mu(G_{\text{res}})$ is the size of a maximum matching in G_{res}. By a similar argument as in the proof of Lemma 6.9, we see that Charlie's edges M^*_g together with Bob's edges $\bigcup_{j\in[r_g]} M^*_{I^*,j}$ constitute a maximum matching in G_{res}. Hence:

$$\begin{aligned}\mu(G_{\text{res}}) &= |M^*_g| + |\bigcup_{j\in[r_g]} M^*_{I^*,j}| \\ &= 2\cdot(n_g - r_g)\cdot(2n_\ell - r_\ell) + r_g \cdot 2 \cdot (n_\ell - r_\ell) \\ &= (2+4\gamma)\cdot n_g \cdot n_\ell \ .\end{aligned}$$

Next, since M is a $(\frac{8}{9}+\epsilon)$-approximation, by Lemma 6.9, we obtain

$$|M| \geq (\frac{8}{9}+\epsilon)(\frac{9}{4} - \frac{\epsilon}{4})\cdot n_g \cdot n_\ell \ .$$

Then, combining the previous inequality with Inequality 6.1 implies

$$\begin{aligned}|M_s| &\geq (\frac{8}{9}+\epsilon)(\frac{9}{4} - \frac{\epsilon}{4})\cdot n_g \cdot n_\ell - \mu(G_{\text{res}}) \\ &= (\frac{8}{9}+\epsilon)(\frac{9}{4} - \frac{\epsilon}{4})\cdot n_g \cdot n_\ell - (2+4\gamma)\cdot n_g \cdot n_\ell \\ &\geq (2\epsilon - \frac{\epsilon^2}{4} - 4\gamma)\cdot n_g \cdot n_\ell \geq \epsilon \cdot n_g \cdot n_\ell \ ,\end{aligned}$$

where the last inequality holds whenever $\epsilon \leq 1$ and $\gamma \leq \frac{1}{8}\epsilon$. This concludes the proof. □

By Lemma 6.10, we know that the $(\frac{8}{9}+\epsilon)$-approximate algorithm $\mathcal{A}$ outputs at least $\epsilon \cdot n_g \cdot n_\ell$ edges from the special matchings $M_\epsilon(I^*, j, K'[I^*, j])$, for every j, when it succeeds. In our subsequent analysis, however, we consider an arbitrary set of exactly $\epsilon \cdot n_g \cdot n_\ell$ of these edges from the output of $\mathcal{A}$, which Charlie can identify using $K[I^*, j]$, for all j.

Lemma 6.11. *For any $j \in [r_g]$ and $\varphi \in [r_\ell]$, the edge $(I^*, \sigma^{r_g}(j), K'[I^*, \sigma^{r_g}(j)], \sigma^{r_\ell}(\varphi))$ is identified from the output of $\mathcal{A}$ with probability*

$$\begin{aligned} p_1 &= \frac{\epsilon}{(1-\epsilon)\cdot(\frac{1}{2}-\gamma)^2} \pm o(1) && \text{if } X''[I^*, j, K[I^*, j]][\varphi] = 1 \ , \text{ and} \\ p_0 &= (1-2\epsilon)\cdot p_1 && \text{if } X''[I^*, j, K[I^*, j]][\varphi] = 0 \ . \end{aligned}$$

Proof. Recall first that, in the construction of graph G, Alice deletes her edges based on the uniform random bit string X''. In particular, the edge $e = (I^*, \sigma^{r_g}(j), K'[I^*, \sigma^{r_g}(j)], \sigma^{r_\ell}(\varphi))$,

which is the $\sigma^{r_\ell}(\varphi)$th edge of the matching $M(I^*, \sigma^{r_g}(j), K'[I^*, \sigma^{r_g}(j)])$ (before Alice's deletions), is *not* deleted (survives) with probability 1 if $X''[I^*, j, K[I^*, j]][\varphi] = 1$ and with probability $1 - 2\epsilon$ if $X''[I^*, j, K[I^*, j]][\varphi] = 0$. Now, conditioned on the event that edge e survives, i.e., $e \in M_\epsilon(I^*, \sigma^{r_g}(j), K'[I^*, \sigma^{r_g}(j)])$, the probability that e is in the output of $\mathcal{A}$ is independent of the value of $X''[I^*, j, K[I^*, j]][\varphi]$. Thus, we can treat both cases in the same way, which further implies that $p_0 = (1 - 2\epsilon) \cdot p_1$.

Suppose that $e = (I^*, \sigma^{r_g}(j), K'[I^*, \sigma^{r_g}(j)], \sigma^{r_\ell}(\varphi))$ survives. The uniform random permutations make e equally likely to be any of the edges (I^*, j', k', φ'), for all j', k', φ'; however, by construction, Bob's edges $\bigcup_{j \in [r_g]} M^*_{I^*, j}$ ensure that e is an edge of the special matchings $M_\epsilon(I^*, j', K'[I^*, j'])$, for all j'. Therefore, e is a uniform edge of the special matchings. Since Charlie identifies exactly $\epsilon \cdot n_g \cdot n_\ell$ edges of the special matchings from the output of $\mathcal{A}$, the probability that e is identified given it survives is

$$
\begin{aligned}
p &= \frac{\epsilon \cdot n_g \cdot n_\ell}{|\bigcup_{j \in [r_g]} M_\epsilon(I^*, j, K'[I^*, j])|} \\
&= \frac{\epsilon \cdot n_g \cdot n_\ell}{(1-\epsilon) \cdot r_g \cdot r_\ell \pm o(r_g \cdot r_\ell)} && \text{(by Lemma 6.8)} \\
&= \frac{\epsilon \cdot n_g \cdot n_\ell}{(1-\epsilon) \cdot (\frac{1}{2} - \gamma)^2 \cdot n_g \cdot n_\ell \pm o(n_g \cdot n_\ell)} && \text{(by Proposition 6.7)} \\
&= \frac{\epsilon}{(1-\epsilon) \cdot (\frac{1}{2} - \gamma)^2} \pm o(1) \ .
\end{aligned}
$$

Finally, the probability that edge e survives *and* it is identified in the output of $\mathcal{A}$ is then $p_1 := p$ if $X''[I^*, j, K[I^*, j]][\varphi] = 1$ and $p_0 := (1 - 2\epsilon) \cdot p$ if $X''[I^*, j, K[I^*, j]][\varphi] = 0$. □

We are now ready to prove our main result:

Proof of Theorem 6.1. Let $\mathcal{A}'$ be the two-pass $(\frac{8}{9} + \epsilon)$-approximation streaming algorithm with constant-error as in the statement of the theorem. We observe that, by running $\mathcal{A}'$ $\Theta(\log n)$ times in parallel and outputting the largest matching computed, we obtain an algorithm $\mathcal{A}$ with the same approximation factor that succeeds with probability $1 - \frac{1}{\text{poly}(n)}$ and uses a factor $\Theta(\log n)$ more space than $\mathcal{A}'$.

Consider now the protocol π_{HS} obtained from $\mathcal{A}$ as described in the beginning of this section. Recall that in π_{HS}, the algorithm $\mathcal{A}$ is run $R = \Theta(\frac{1}{\epsilon} \cdot \log^2(n_g \cdot n_\ell))$ times in parallel. By a union bound, we have that all runs succeed with probability at least $1 - \frac{1}{\text{poly}(n)} \cdot R = 1 - \frac{1}{\text{poly}(n)}$, and we condition on the event that this happens from now on.

Now, consider any relevant bit with value 1, i.e., $X''[I^*, j, K[I^*, j]][\varphi] = 1$ for some j and φ. By Lemma 6.11, its corresponding edge is reported in the output of $\mathcal{A}$ with probability $p_1 = \frac{\epsilon}{(1-\epsilon) \cdot (\frac{1}{2} - \gamma)^2} \pm o(1) \geq \epsilon$, and thus is expected to be reported $\mu_1 := p_1 \cdot R \geq C \cdot \log^2(n_g \cdot n_\ell)$ times, for some constant C. By a standard concentration bound, we have that the edge is

reported $\mu_1 \pm o(\mu_1)$ times with probability $1 - \frac{1}{\text{poly}(n_g \cdot n_\ell)}$. Similarly, by Lemma 6.11, the edge of a relevant bit with value 0 is expected to be reported $\mu_0 := p_0 \cdot R = (1-2\epsilon)\mu_1 \geq C \log^2(n_g \cdot n_\ell)$ times, and, by concentration bounds, the bit is reported $\mu_1(1-2\epsilon) \pm o(\mu_1)$ times with probability $1 - \frac{1}{\text{poly}(n_g \cdot n_\ell)}$.

By counting how often the edges corresponding to a relevant bit are reported in the output matchings of the parallel runs, second-round Charlie can thus distinguish between the two cases (e.g., by comparing the count to the threshold $\mu_1(1-\epsilon)$) and correctly predict the value $X''[I^*, j, K[I^*, j][\varphi]]$, for some j and φ, with probability $1 - \frac{1}{\text{poly}(n_g \cdot n_\ell)}$. Undoing the XOR operation then yields the desired bit $X[I^*, j, K[I^*, j]][\varphi]$. By a union bound over all relevant bits, i.e., $X''[I^*, j, K[I^*, j]][\varphi]$, for all j and φ, Charlie correctly outputs all bit strings $X[I^*, j, K[I^*, j]]$, for all j, with probability $1 - \frac{1}{\text{poly}(n_g \cdot n_\ell)}$, and further union bounding over the event that all runs of $\mathcal{A}$ succeed shows that the protocol π_{HS} succeeds with probability at least $1 - \frac{1}{\text{poly}(n_g \cdot n_\ell)}$.

Let $|\mathcal{A}|$ be the maximum space used by the algorithm. Protocol π_{HS} communicates $\mathcal{A}$'s memory state R times and has an additional $O(\log t_g + r_g \cdot \log t_\ell)$ bits of overhead communication since first-round Charlie forwards I^* and second-round Bob forwards $K[I^*, j]$, for all j. This yields a protocol π_{HS} with communication cost $O(\frac{1}{\epsilon} \cdot \log^2(n_g \cdot n_\ell) \cdot |\mathcal{A}| + \log t_g + r_g \cdot \log t_\ell)$. Then, by Theorem 6.3 and since ϵ is constant, we have that

$$|\mathcal{A}| = \Omega\left(\frac{\min\{t_g \cdot r_g, r_g \cdot t_\ell \cdot r_\ell\}}{\log^2(n_g \cdot n_\ell)}\right) .$$

It remains to find parameters n_g and n_ℓ such that $\min\{t_g \cdot r_g, r_g \cdot t_\ell \cdot r_\ell\}$ is sufficiently large.

Recall that the construction of G in the above reduction relies on a $(2n_g)$-vertex RS-graph and a $(2n_\ell)$-vertex RS-graph as in Proposition 6.7. These RS-graphs do not exist for all choices of n_g and n_ℓ; however, we show that there exist suitable choices. In particular, we argue the following:

First, there exist constants c and n_0 such that the RS-graph on $2N$ vertices described in Proposition 6.7 has at least $N^{1+\frac{c}{\log\log N}}$ edges, for every $N \geq n_0$. Let $n_\ell \geq n_0$ be large enough such that there exists an RS-graph as in Proposition 6.7 with $2n_\ell$ vertices. Let $n'_g = n_\ell^{\frac{2}{c}\log\log(n_\ell)-1} \cdot \log^3(n_\ell)$, which, by Proposition 6.7, implies that there exists an RS-graph on $2n_g$ vertices such that $n_g \geq \frac{n'_g}{\log^2(n'_g)} \geq n_\ell^{\frac{2}{c}\log\log(n_\ell)-1}$, where the last inequality holds since $\frac{\log^3 n_\ell}{\log^2 n'_g} \geq 1$ for large enough n_ℓ.

Each graph G in the reduction has $n = O(n_g \cdot n_\ell) = O(n' \log^3 n_\ell)$ vertices where $n' = n_\ell \cdot n_\ell^{\frac{2}{c}\log\log(n_\ell)-1} = n_\ell^{\frac{2}{c}\log\log(n_\ell)}$ and thus $n_\ell = n'^{\frac{c}{2\log\log n_\ell}}$. It then follows that

$$\begin{aligned} t_g \cdot r_g &\geq n_g^{1+\frac{c}{\log\log(n_g)}} \geq n_\ell^{(\frac{2}{c}\log\log(n_\ell)-1)\cdot(1+\frac{c}{\log\log(n_g)})} = n'^{\frac{c}{2\log\log(n_\ell)}(\frac{2}{c}\log\log(n_\ell)-1)\cdot(1+\frac{c}{\log\log(n_g)})} \\ &= n'^{1+\Omega(\frac{1}{\log\log(n_\ell)})} = n^{1+\Omega(\frac{1}{\log\log n})} , \end{aligned}$$

using the fact that $\log\log(n_g) = \log\log(n_\ell) + o(\log\log(n_\ell))$. We further have that

$$r_g \cdot t_\ell \cdot r_\ell \geq r_g \cdot n_\ell^{1+\frac{c}{\log\log(n_\ell)}} \geq \Theta(n_g) \cdot n_\ell^{1+\frac{c}{\log\log n'}} = n'^{1+\Omega(\frac{1}{(\log\log n')^2})} = n^{1+\Omega(\frac{1}{(\log\log n)^2})} .$$

Finally, it follows that $\min\{t_g \cdot r_g, r_g \cdot t_\ell \cdot r_\ell\} \geq n^{1+\Omega(\frac{1}{(\log\log n)^2})}$, and thus the constant-error algorithm $\mathcal{A}'$ uses space $|\mathcal{A}'| \geq \frac{|\mathcal{A}|}{\Theta(\log n)} \geq \frac{\min\{t_g \cdot r_g, r_g \cdot t_\ell \cdot r_\ell\}}{\Theta(\log^3 n)} = n^{1+\Omega(\frac{1}{(\log\log n)^2})}$. □

6.5 Conclusion

In this chapter, we gave a lower bound on the space required by two-pass $(\frac{8}{9}+\epsilon)$-approximation streaming algorithm for MBM and showed that each such algorithm requires space $n^{1+\Omega(\frac{1}{(\log\log n)^2})}$.

Putting our lower bound in perspective with the best currently known two-pass semi-streaming algorithm for MBM by Konrad [Kon18b] (see also [KN21, BKSW23]), we now know that the best achievable approximation factor in the two-pass semi-streaming model is in the range $[2-\sqrt{2}, \frac{8}{9}] \approx [0.585, 0.889]$. Narrowing this gap any further, both from the algorithmic and the lower bound perspectives, is an exiting open problem. Additionally, we conjecture that the ideas and techniques of our lower bound could be used to make further progress in regards to proving a better two-pass lower bound for MBM and proving an unconditional multi-pass lower bound for MBM. Perhaps more interestingly, our techniques have already inspired subsequent work that proves an optimal (up to constant factors) semi-streaming lower bound for maximal independent set [AKNS24].

A better two-pass lower bound for MBM. Our hard graph construction for two passes can be seen as a generalisation of the hard graph construction in [GKK12] that rules out better than (2/3)-approximations in one pass. We believe that the same ideas and techniques can be used to generalise the more intricate hard bipartite graph construction in [Kap13] or [Kap21], either of which would obtain an improved two-pass lower bound for semi-streaming MBM. In particular, the construction in [Kap21] that rules out better than $(1/(1+\ln 2))$-approximations in one pass could be used to rule out better than (≈ 0.833)-approximations in two passes.

An unconditional multi-pass lower bound for MBM. It is possible to add further levels to the hierarchy of RS-graphs in our lower bound construction so as to obtain lower bounds for semi-streaming algorithms that use more than two passes. More specifically, the idea is to use an RS-graph as a global graph in the same way as our two-pass construction and instead use our hard two-pass bipartite graph construction as the local graphs. We believe this construction to be hard in three passes for MBM, ruling out better than (26/27)-approximations in semi-streaming space. This can be repeated in a similar manner to obtain hard graph

constructions for p passes, ruling out better than $(3^p - 1/3^p)$-approximations in semi-streaming space, that is, ruling out a better than $(1-\varepsilon)$-approximation in semi-streaming space using $o(\log(1/\varepsilon))$ passes, which is currently only conditionally proven in [AS23a]. The techniques using the information-cost tradeoff result for Index that we develop for proving our two-pass lower bound, however, are limited to two passes due to an independence property that does not hold when considering even just three passes.

Developments since the conference version of this chapter [KN24]. Very recently, Assadi et al. [AKNS24] proved the first and optimal (up to constant factors) multi-pass lower bound for semi-streaming maximal independent set (MIS) by obtaining a hierarchical graph construction that is inspired by our work. They give a generalisation of RS graphs that they use to extend our ideas for constructing hierarchical graphs to allow for k-colourable graphs, rather than just bipartite (or 2-colourable) graphs as in our case – this is a key requirement for MIS as bipartite graphs are trivially easy. By adding many levels (in a similar, but slightly more involved manner, as we discussed in the previous paragraph), they obtain hard graph constructions for MIS in multiple passes. Finally, they prove the hardness of the constructions using a novel round-elimination argument using message compression, showing that $\Omega(\log\log n)$ passes is required by any semi-streaming algorithm and matching the $O(\log\log n)$-pass algorithm (up to constant factors) [ACG+15]. Their round-elimination technique is also sufficient to prove our conjectured unconditional lower bound for multi-pass MBM.

7 Insertion-Deletion Approximate MVC

This chapter has been published as the following work:

- [NS22] (**student-only**): Kheeran K. Naidu and Vihan Shah. Space Optimal Vertex Cover in Dynamic Streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022.*

Abstract

We optimally resolve the space complexity for the problem of finding an α-approximate minimum vertex cover (αMVC) in insertion-deletion (or, equivalently, dynamic) graph streams. We give a randomised algorithm for αMVC which uses $O(n^2/\alpha^2)$ bits of space matching Dark and Konrad's lower bound [CCC 2020] up to constant factors. By computing a random greedy matching, we identify 'easy' instances of the problem which can trivially be solved by returning the entire vertex set. The remaining 'hard' instances, then have sparse induced subgraphs which we exploit to get our space savings and solve αMVC.

Achieving this type of optimality result is crucial for providing a complete understanding of a problem, and it has been gaining interest within the insertion-deletion streaming community. For connectivity, Nelson and Yu [SODA 2019] improved the lower bound showing that $\Omega(n \log^3 n)$ bits of space is necessary while Ahn, Guha, and McGregor [SODA 2012] have shown that $O(n \log^3 n)$ bits is sufficient. For finding an α-approximate maximum matching, the upper bound was improved by Assadi and Shah [ITCS 2022] showing that $O(n^2/\alpha^3)$ bits is sufficient while Dark and Konrad [CCC 2020] have shown that $\Omega(n^2/\alpha^3)$ bits is necessary. The space complexity, however, remains unresolved for many other insertion-deletion streaming problems where further improvements can still be made.

7.1 Introduction

Graph streaming is a setting in which a graph is specified by a sequence of edges, typically in arbitrary order. It is particularly useful for processing massive graphs where having random access to the edges of the graph is either impossible or computationally infeasible.

Research in this area began with *insertion-only streams*, where the stream is made up of a sequence of edge insertions only. In their seminal work, Feigenbaum, Kannan, McGregor, Suri, and Zhang [FKM+04] showed that for many problems including minimum spanning tree, connectivity, and bipartiteness, $\Omega(n)$ bits of space is necessary and $O(n \log n)$ bits is sufficient for any n-vertex graph. This logarithmic gap was often overlooked and deemed not important when proving optimality for graph problems, but it left unresolved the question of whether the logarithmic factor was required for simply storing edges or if other techniques could remove it. About a decade later, Sun and Woodruff [SW15] showed that the logarithmic factor was indeed necessary by improving the lower bounds to $\Omega(n \log n)$ bits, asymptotically matching the upper bounds up to constant factors.

Insertion-deletion streams, which allow for sequences of both edge insertions and deletions, prove to be more difficult. Edges that arrive in the stream are not necessarily in the final graph as they may later be deleted. In fact, it is well-known in the community that it is impossible to deterministically return a single edge of a dense graph without storing all of its edges. As a result, almost all insertion-deletion streaming algorithms rely on counters which use $O(\log n)$ bits of space or they rely on L_0-sampling which optimally uses $\Theta(\log^3 n)$ bits of space[1] [JST11, KNP+17]. In essence, counters are used to solve the problem of determining whether an edge is present in an edge induced subgraph [DK20] (see also [CCE+16]), whereas L_0-sampling also returns the identity of a uniform random edge if one is present [AGM12a, AGM12b, Kon15, CCE+16, AKLY16, ACK19b, Kon21, KK22, AS22]. A notable exception includes spectral sparsification [KLM+14, KMM+20] which relies on L_2-heavy-hitters (non-uniform sampling).

Resolving the space complexity up to constant factors for insertion-deletion streaming problems has continued to be an elusive task. Ahn, Guha, and McGregor [AGM12a] gave an algorithm for connectivity using $O(n \log^3 n)$ bits of space, and for several years, the best known lower bound was the insertion-only bound of $\Omega(n \log n)$ bits [SW15]. However, in 2019, Nelson and Yu [NY19] improved the lower bound to $\Omega(n \log^3 n)$ bits in the insertion-deletion streaming setting. To the best of our knowledge, this is the only problem in this setting which has space bounds that prove the necessity of the $\Theta(\log^3 n)$ overhead of randomly sampling an edge (using L_0-sampling). The approximate minimum cut problem which has a $\Omega(n \log^3 n)$ bit lower bound [NY19] (and a $O(n \log^4 n)$ bit upper bound [AGM12a]) similarly shows that logarithmic factors are necessary. A perhaps more surprising result was the recent progress on

[1] This optimal space bound applies when the probability of success is at least $1 - \frac{1}{\text{poly}(n)}$.

α-approximate maximum matching (αMM). The lower bound of $\Omega(\alpha^3 \cdot n^2)$ bits [DK20] (see also [AKLY16]) and the previous upper bound of $O(\alpha^3 \cdot n^2 \cdot \log^4 n)$ bits [AKLY16, CCE+16] seem to indicate that the logarithmic overhead of sampling an edge is required. However, Assadi and Shah [AS22] improved the upper bound to $O(\alpha^3 \cdot n^2)$ bits showing that this is not the case. On the other hand, for problems such as vertex cover [DK20], dominating set [KK22], and spectral sparsification [KMM+20], their space bounds have a gap of logarithmic factors, and therefore further improvements can still be made.

Our Results. In this work, we optimally resolve the space complexity up to constant factors for the problem of finding an α-approximate minimum vertex cover (αMVC) in an insertion-deletion graph stream[2]. In particular, we improve the upper bound to $O(n^2/\alpha^2)$ bits, matching the $\Omega(n^2/\alpha^2)$ bits lower bound [DK20] and showing that the logarithmic overhead is not required. Our main result is the following:

Theorem 7.1 (cf. Theorem 8)**.** *There exists a randomised insertion-deletion streaming algorithm for αMVC that succeeds with high probability and uses $O(n^2/\alpha^2)$ bits of space for any $\alpha \leq n^{1-\delta}$ where $\delta > 0$.*

Previous Work. It has been shown by Dark and Konrad [DK20] that $\Omega(n^2/\alpha^2)$ bits is necessary for αMVC. They also gave a simple deterministic algorithm which uses $O(n^2/\alpha^2 \cdot \log \alpha)$ bits of space, matching the lower bound up to logarithmic factors. Their algorithm arbitrarily partitions the vertex set into n/α groups of size α and uses counters, which introduces the logarithmic overhead, to maintain the number of edges between each of the $\Theta(n^2/\alpha^2)$ pairs of vertex groups. The solution follows by computing a group-level minimum vertex cover, and then returning the vertices of the covering groups.

Main Techniques. We improve the approach of Dark and Konrad [DK20] by additionally computing a supporting random Greedy matching and randomly partitioning the vertex set into n/α groups, effectively using randomisation to reduce the space required. The random Greedy matching returned is either large enough to imply a trivial solution for αMVC ('easy' case) or implies sparseness properties of the residual subgraph induced by the unmatched vertices ('hard' case). To solve the 'hard' cases, we use the sparseness properties and the random partitioning to argue that there are only $O(1)$ many edges between each pair of vertex groups in the residual subgraph. Therefore, storing edge counters for each of the $\Theta(n^2/\alpha^2)$ many pairs, as done by Dark and Konrad [DK20], now requires only $O(n^2/\alpha^2)$ bits of space in total.

Sampling Strategies. The sparseness properties (of the residual subgraph) implied are reliant on the method of randomly sampling edges from the graph. Uniformly sampling from

[2]Recall that, for a minimisation problem such as MVC, an α-approximation is a solution that is no larger than α times an optimal solution where $\alpha > 1$.

the edge set only implies sparseness properties sufficient for a small range of α since it is skewed to sampling high degree vertices. On the other hand, non-uniform sampling – sampling from the neighbourhood of a random set of vertices, coined *neighbourhood edge sampling* by Assadi and Shah [AS22] – is less biased towards high degree vertices and implies the necessary sparseness properties for the full range of α. Indeed, Assadi and Shah [AS22] also use the approach of computing a Greedy matching on non-uniformly sampled edges to identify the 'easy' and 'hard' instances of αMM. However, for αMVC, our 'easy' and 'hard' instances differ from those of αMM, so we require different guarantees. Furthermore, we use different techniques for solving the 'hard' instances.

Further Related Work. Resolving the space complexity up to constant factors has also been achieved for non-graph problems in the general data streaming setting. For instance, Braverman, Katzman, Seidell, and Vorsanger [BKSV14] gave an upper bound for finding a constant factor approximation to the k-th frequency moment in constantly many passes that matches the lower bound of Woodruff and Zhang [WZ12]. Price and Woodruff [PW13] showed a lower bound for any adaptive sparse recovery scheme that matches the upper bound of Indyk, Price, and Woodruff [IPW11]. Graph problems in other streaming settings have also been studied. For example, the settings which allow multiple passes over the stream [KMM12, Kon18b, ACK19a, AKSY20, KN21, Ass22, AJJ+22], have a random arrival order [KMM12, ABB+19, Ber20], or have highly structured deletions via a sliding window [CMS13, CS14, BdBM21] have been considered. See the work by McGregor [McG14] for an excellent survey on graph streaming algorithms.

Outline. We begin in Section 7.2 with some important notation and tools which we will later use. In Section 7.3, we discuss the guarantees required from a random Greedy matching for αMVC. In Section 7.4, we present and analyse our algorithm that proves Theorem 7.1. Then, we conclude in Section 7.5.

7.2 Preliminaries

For any n-vertex graph $G = (V, E)$, let $\mu(G)$ be the size of the maximum matching of the graph, let $V^*(G)$ be a minimum vertex cover, and let $\text{opt}(G)$ be its size. We will simply use μ, V^* or opt if the graph is clear from context. For any subset of edges $F \subseteq E$, we denote the set of their endpoints by $V(F)$. For any subgraph H of G and vertex $v \in V$, we use $N_H(v)$ to denote the neighbourhood of v in H .

The graph G may be specified as an insertion-deletion graph stream[3] $\sigma = (\sigma_1, \sigma_2, ..., \sigma_N)$ such that $\sigma_j = (i_j, \Delta_j)$ where $i_j \in [m]$ for $m = \binom{n}{2}$ and $\Delta_j \in \{1, -1\}$ (insertions or deletions).

[3] An insertion-deletion graph stream is a special case of the strict turnstile data streaming model [Mut05] where we consider only bit-vectors which represent the edges of a graph.

Note that edges may only be deleted if they have previously been inserted. Additionally, the stream must produce a vector $\mathbf{vec}(E) \in \{0,1\}^m$ that defines the edge set E, i.e., the i^{th} entry of the vector indicates the presence of the edge indexed by $i \in [m]$.

In our work, we will rely on limited independence hash functions to reduce the space complexity of our algorithm. Roughly speaking, a hash function sampled from a family of k-wise independent hash functions behaves like a totally random function when considering at most k elements. For simplicity, when we mention a k-wise independent hash function, we will mean a hash function sampled from a family of k-wise independent hash functions. We use the following standard result for k-wise independent hash functions.

Proposition 7.2 ([MR95])**.** *For all integers $n, m, k \geq 2$, there is a family of k-wise independent hash functions $\mathcal{H} = \{h : [n] \to [m]\}$ such that sampling and storing a function $h \in \mathcal{H}$ takes $O(k \cdot (\log n + \log m))$ bits of space.*

We shall also use the following concentration result on an extension of Chernoff-Hoeffding bounds for k-wise independent hash functions.

Proposition 7.3 ([SSS93])**.** *Suppose h is a k-wise independent hash function and $X_1, \ldots, X_m$ are m random variables in $\{0,1\}$ where $X_i = 1$ iff $h(i) = 1$. Let $X := \sum_{i=1}^{m} X_i$. Then, for any $\varepsilon > 0$,*

$$\Pr\left(|X - \mathbb{E}[X]| \geq \varepsilon \cdot \mathbb{E}[X]\right) \leq \exp\left(-\min\left\{\frac{k}{2}, \frac{\varepsilon^2}{4+2\varepsilon} \cdot \mathbb{E}[X]\right\}\right).$$

Finally, we will use the following sketching tool for insertion-deletion graph streams to test the size of the neighbourhood of a subset of vertices.

Proposition 7.4 ([AS22])**.** *Let $a \geq b \geq 2$ be known integers. Consider a n-vertex graph $G = (V, E)$ specified in an insertion-deletion stream and let $S \subseteq V$ be a known set. Then, given a set $T \subseteq V$ of size at most a at the end of the stream, there exists a randomised algorithm that returns "Yes" if $|N_G(S) \backslash T| \geq b$ or "No" if $|N_G(S) \backslash T| \leq \frac{1}{2} \cdot b$, uses $O(\frac{a}{b} \cdot \log^3 n)$ bits of space, and succeeds with probability at least $1 - n^{-3}$. We denote one such algorithm as $\mathcal{ALG}_{NT}(S; a, b)$.*

7.3 Sampling Strategies for Random Greedy Matchings

In this section, we discuss and present the tool that we use to either find a large matching or show that the residual subgraph induced by the unmatched vertices is sparse.

This approach was also used in Assadi and Shah's recent work for αMM [AS22] to identify 'easy' and 'hard' instances of the problem. For αMVC, the 'easy' case is finding a large enough matching to imply that we can trivially return the entire vertex set to solve the problem. The

'hard' case is when we get a sparse residual subgraph, which is where our main savings in space come from. Identifying these cases can be accomplished by computing a Greedy matching on randomly sampled edges of the graph.

Uniformly sampling as many edges as possible from a n-vertex graph (using L_0-sampling) without exceeding $O(n^2/\alpha^2)$ bits of space followed by computing a Greedy matching implies sparseness properties based on an already known maximum degree bound of the residual subgraph induced by the unmatched vertices [ACG+15, Kon18a, GKMS19]. Intuitively, uniform sampling is skewed towards sampling edges incident to high degree vertices. Hence, a Greedy matching either matches these high degree vertices or matches many of its neighbours (decreasing their residual degree), and regardless of the size of the matching found, this gives a poly(α) max degree bound in the residual graph. Furthermore, we can show that this also bounds the average degree (even when a small matching is found) since a worst-case instance[4] practically has all vertices in the residual subgraph with max degree. This degree bound, however, is only sufficient for solving αMVC for any $\alpha \ll n^{\frac{1}{3.5}}$.

Non-uniformly sampling the edges using neighbourhood edge sampling followed by Greedy, as done by Assadi and Shah [AS22], proves to give better sparseness properties, and thus a better average degree bound[5]. The benefit of neighbourhood edge sampling is that it biases away from sampling high degree vertices. Furthermore, when a small Greedy matching is found, the implication is that the residual subgraph is sparse. Therefore, the average degree bound is sufficient for solving the 'hard' case of αMVC for the full range of α.

As previously mentioned, Assadi and Shah's algorithm called Match-or-Sparsify [AS22], does exactly this, although its guarantees are not sufficient for our purposes. Hence, we first discuss their algorithm, and then explain the alterations we make.

Match-or-Sparsify. For some parameter $\beta \leq n$, Assadi and Shah's Match-or-Sparsify$_\beta$ algorithm non-uniformly samples edges using space $O(\beta^2/\alpha^3)$ bits, and then computes a Greedy matching from them. They give an intricate analysis to show that their algorithm either finds a large matching of size at least $\beta/8\alpha$ or implies that the residual subgraph has at most $20 \cdot \beta \cdot \log^4 n$ edges [AS22, Lemma 16]. Unlike uniform sampling, the residual properties (sufficiently) only hold when the matching is small – a key property exploited in their analysis. Additionally, in order for the guarantees to hold, they rely on the assumption that $\beta \geq \alpha^2 \cdot n^\delta$. Informally, when β is set as the size of the maximum matching μ, Match-or-Sparsify$_\mu$ finds a large matching in 'easy' graph cases and a sparse residual subgraph in the 'hard' graph cases. However, μ is not known, so they find a setting of β close to μ by running Match-or-Sparsify$_\beta$ in parallel with

[4] Consider a graph with a large clique on $\Theta(n/\alpha)$ vertices where most the edges are sampled from, and many smaller cliques which assert the guaranteed max degree bounds.

[5] Having an average degree bound is more difficult to work with, but in this case, the bound on the average degree is much smaller than the bound on the max degree in the uniform case.

β as all powers of 2 between 1 and n.

Our Alterations. The first thing to note is that the 'easy' and 'hard' instances for αMM and αMVC are not the same. Consider Match-or-Sparsify$_\beta$ when a large Greedy matching is found. Since at least one endpoint of each matching edge must be in a vertex cover, it implies that opt $\geq \frac{\beta}{8\alpha}$. However, returning a solution to αMVC at this stage can only be of size at most $\Theta(\beta)$, which would not be a trivial solution (the entire vertex set) with $\beta \ll n$. Furthermore, we have no guaranteed sparseness properties since the matching found is large. Hence, instead of needing $\beta \approx \mu$, which requires $\log n$ many runs to find, we only need a single run of Match-or-Sparsify$_n$ (with the parameter β fixed as n). Secondly, their assumption that $\beta \geq \alpha^2 \cdot n^\delta$ implies that $\alpha \leq n^{\frac{1-\delta}{2}}$, but we require it to hold for any $\alpha \leq n^{1-\delta}$. Since we have an additional α factor of space (see [DK20]), we can increase the number of non-uniformly sampled edges to use $O(n^2/\alpha^2)$ bits instead, which allows us to remove the assumption. Finally, the increase in the number of samples also allows us to increase the sparseness guarantees of the residual subgraph by an α factor. Therefore, this altered Match-or-Sparsify$_n$ algorithm, denoted by $\mathcal{ALG}_{MS}$, gives us the following lemma (full proof given in Appendix A.2 for completeness).

Lemma 7.5. *There is a linear sketch for insertion-deletion graph streams that, given any graph $G = (V, E)$ specified via $\mathbf{vec}(E)$, uses $O(n^2/\alpha^2)$ bits of space and with high probability outputs a matching M_{easy} that satisfies at least one of the following conditions for any $\alpha \leq n^{1-\delta}$ and $\delta > 0$:*

- ***Match-case:*** *The matching M_{easy} has at least $\frac{n}{8\alpha}$ edges;*
- ***Sparsify-case:*** *The induced subgraph of G on vertices not matched by M_{easy}, denoted by G_R, has at most $20 \cdot \frac{n}{\alpha} \cdot \log^4 n$ edges.*

7.4 Main Result

In this section, we give an insertion-deletion streaming algorithm for αMVC for any n-vertex graph which implies our main result:

Theorem 7.1 (cf. Theorem 8). There exists a randomised insertion-deletion streaming algorithm for αMVC that succeeds with high probability and uses $O(n^2/\alpha^2)$ bits of space for any $\alpha \leq n^{1-\delta}$ where $\delta > 0$.

Before proceeding, we give the following standard assumption (with reason) which simplifies what we need to prove.

Assumption 7.6. A randomised insertion-deletion streaming $\Theta(\alpha)$-approximation algorithm that uses $O(n^2/\alpha^2)$ bits of space and succeeds on graphs where opt $\geq \frac{n}{\alpha \cdot \log^2 n}$ is sufficient to prove Theorem 7.1.

Reason. Let $\mathcal{A}$ be an algorithm that returns a $(c \cdot \alpha)$-approximation using $O(n^2/\alpha^2)$ bits of space. Run $\mathcal{A}$ with parameter α/c to get an α-approximation which similarly uses $O(n^2/\alpha^2)$ bits.

Then, since we can run $\Theta(1)$ many algorithms which use $O(n^2/\alpha^2)$ bits of space in parallel without asymptotically increasing the space, we run an additional algorithm which detects and outputs a solution for graphs with small opt.

Algorithm for small opt**.** We use the well-known algorithm for finding an exact minimum vertex cover in insertion-deletion graph streams with probability at least $1 - \frac{1}{\text{poly}(k)}$ given the promise that opt $\leq k$ with $k = \frac{n}{\alpha \cdot \log^2 n} \geq n^{\delta/2}$ [CCE+16]. Note that the poly(k) can be made a function of δ to get a success probability of at least $1 - k^{-20/\delta} \geq 1 - n^{-10}$. If opt $< k$, then we get an optimal solution; otherwise, we get a set of vertices of size k which are not necessarily a solution. Thus, we can detect this case by the size of the returned vertex cover being smaller than k. The space taken by the algorithm is $O(k^2 \cdot \log^4 n) = O(n^2/\alpha^2)$ bits and it works for all $\alpha = \omega(1)$ (for $\alpha = \Theta(1)$ we can store the entire graph). □

Algorithm Description. Let $G = (V, E)$ be specified by an insertion-deletion graph stream, $\delta > 0$, and $\alpha \leq n^{1-\delta}$ be the inputs to Algorithm 5. The algorithm, in its pre-processing step, partitions V into n/α groups using a $(10 \cdot \log n)$-wise independent hash function (when the space allows, i.e., for small α, we do this using a uniform random permutation instead), and we later show that all their sizes lie between $\alpha/2$ and 2α with high probability. During the stream, it maintains counters modulo some constant for the number of edges between each pair of groups and (standard) counters for the number of internal edges of each group. In parallel, it computes a random matching M_{easy} using an instance of $\mathcal{ALG}_{MS}$ (Lemma 7.5) and maintains residual neighbourhood size testers for each vertex group using instances of $\mathcal{ALG}_{NT}$ (Proposition 7.4). In the post-processing step, if the matching is of size at least $\frac{n}{8 \cdot \alpha}$, then the entire vertex set is returned. Otherwise, the vertex groups containing any vertex of the matching or any internal edges are entirely picked in the solution – we call these *simple vertex groups*. Next, the remaining vertex groups V_i whose residual neighbourhood is large, $|N_{G_R}(V_i)| = |N_G(V_i) \backslash V(M_{\text{easy}})| \geq n^{\delta/2}$ where $G_R = G[V \backslash V(M_{\text{easy}})]$, are added to the solution – we call these *residual vertex groups*. Finally, among the leftover *clean vertex groups*, the algorithm uses the counters modulo some constant to perform a group-level vertex cover, and then further adds the covering groups to the solution before returning it.

Definition 7.7 (Simple Vertex Groups)**.** We say that a vertex group V_i is *simple* if any of its vertices are matched by M_{easy} or it has at least one internal edge, i.e., $|V_i \cap V(M_{\text{easy}})| > 0$ or $C_i > 0$.

Algorithm 5 Optimal Insertion-Deletion Vertex Cover

Input: An insertion-deletion graph stream σ for a n-vertex graph $G = (V, E)$, a small constant $\delta > 0$, and a positive integer $\alpha \leq n^{1-\delta}$
Output: A vertex cover V_C of G

Pre-processing:

1: Initialise $\mathbf{M}$ to be an instance of $\mathcal{ALG}_{MS}$ (Lemma 7.5)
2: Randomly partition V into groups $V_1, V_2, ..., V_{\frac{n}{\alpha}}$ having size in $[\alpha/2, 2\alpha]$
3: For each group V_i, initialise $\mathbf{N}_i$ to be an instance of $\mathcal{ALG}_{NT}(V_i; a, b)$ (Proposition 7.4) with $a = n/\alpha$ and $b = n^{\delta/2}$
4: Set $c = 15/\delta$

Processing the stream:

5: Update $\mathbf{M}$ and each $\mathbf{N}_i$ using σ
6: For every pair of groups V_i and V_j, store a counter $C_{i,j}$ for the number of edges between them modulo c
7: For every group V_i, store a counter C_i for the number of internal edges

Post-processing:

8: Let M_{easy} be the matching returned by $\mathbf{M}$
9: **if** M_{easy} has at least $\frac{n}{8\cdot\alpha}$ edges **then return** V
10: Let V_C be the union of all groups V_i containing a vertex of M_{easy} or with $C_i > 0$
11: Add to V_C all remaining vertex groups V_i where $\mathbf{N}_i$ returns "Yes" when $T = V(M_{\text{easy}})$
12: Consider the multi-graph G' obtained by contracting the vertices of each remaining vertex group V_i into a single vertex v_i where $C_{i,j}$ represents the multiplicity modulo c of each edge (v_i, v_j) in G'
13: Greedily compute a vertex cover V'_C of G'
14: For all $v_i \in V'_C$, add vertex group V_i to V_C
15: **return** V_C

Definition 7.8 (Residual Vertex Groups)**.** We say that a vertex group V_i is *residual* if it is not simple and has a large residual neighbourhood, i.e., $|N_{G_R}(V_i)| \geq n^{\delta/2}$.

Definition 7.9 (Clean Vertex Groups)**.** We say that a vertex group V_i is *clean* if it is not simple or residual, i.e., $|V_i \cap V(M_{\text{easy}})| = 0$, $C_i = 0$ and $|N_{G_R}(V_i)| < n^{\delta/2}$.

Note that throughout the subsequent analysis of Algorithm 5, all results succeed with high probability. Hence, at any point, we can do a simple union bound to show that they all hold with high probability. As such, we condition on this event here to avoid explicitly doing so during the analysis.

Let G be the input graph of the algorithm. We begin the analysis with the following observation: If G contains a matching of size at least $\frac{n}{8\cdot\alpha}$, then V is a valid $(8\cdot\alpha)$-approximation of a minimum vertex cover V^* since at least one endpoint of a matching edge must be in a valid vertex cover. Therefore, if the condition of Line 9 is satisfied, the algorithm terminates and

the solution is a valid $\Theta(\alpha)$-approximation ('easy' graph instances). Otherwise, the algorithm progresses with $|M_{\text{easy}}| < \frac{n}{8\cdot\alpha}$, i.e., the sparsify-case of Lemma 7.5 ('hard' graph instances). This implies that the residual subgraph G_R is sparse with at most $20 \cdot \frac{n}{\alpha} \cdot \log^4 n$ many edges. As such, we need to prove that we also get a $\Theta(\alpha)$-approximation in the sparsify-case.

We highlight here that the algorithm adds vertex groups to the solution for various reasons, which are determined by whether it is a simple, residual, or clean vertex group (see Definitions 7.7 to 7.9). Hence, we proceed with the analysis of the sparsify-case by considering these different types of vertex groups separately.

Simple Vertex Groups. Let $\mathcal{I}_s$ be the index set of the simple vertex groups. We argue that there are not too many of these, so we can add all of them to the solution.

Claim 7.10. *The number of simple vertex groups $|\mathcal{I}_s|$ is at most $2 \cdot \text{opt}(G)$.*

Proof. Each edge of the matching M_{easy} can cause up to two vertex groups to be classified as simple; however, they must have at least one vertex of V^* since at least one endpoint of every matching edge must be in V^*. Therefore, for every two groups classified as simple in this way, there is at least one vertex of V^* in their union. On the other hand, a group could also be classified as simple if it contains an internal edge, where one of its endpoints must be in V^*. Hence, for each group classified as simple in this way, there is at least one vertex of V^* in it. Then, it follows that the number of simple vertex groups must be at most $2 \cdot |V^*| = 2 \cdot \text{opt}$. □

Residual Vertex Groups. Let $\mathcal{I}_r$ be the index set of the residual vertex groups. Recall that any residual vertex group must have at least $n^{\delta/2}$ many residual neighbours. We note, however, that due to the guarantees of the neighbourhood size tester algorithm $\mathcal{ALG}_{NT}$ (see Proposition 7.4), there are some misclassifications, so some residual vertex groups are also of size between $\frac{1}{2} \cdot n^{\delta/2}$ and $n^{\delta/2}$. This will not be an issue, and moving forward, when we mention residual vertex groups, we assume that this includes the misclassifications. Now, we argue that there are not too many residual vertex groups, so we can add them all to the solution.

Claim 7.11. *The number of residual vertex groups $|\mathcal{I}_r|$ is at most $\text{opt}(G)$ with high probability.*

Proof. We have that $|V(M_{\text{easy}})|$ is at most $\frac{n}{4\cdot\alpha}$ and G_R has at most $20 \cdot \frac{n}{\alpha} \cdot \log^4 n$ many edges. As such, G_R has $n - |V(M_{\text{easy}})| \geq \frac{n}{2}$ vertices, and the average degree of a vertex in G_R is at most $20 \cdot \frac{n}{\alpha} \cdot \log^4 n \cdot \frac{2}{n} = \frac{40 \log^4 n}{\alpha}$. Since each non-simple vertex group V_i is fully contained in G_R and has at most 2α vertices, we have that $\mathbb{E}\left[|N_{G_R}(V_i)|\right] \leq \frac{40 \log^4 n}{\alpha} \cdot 2\alpha = 80 \log^4 n$. Then, it follows by Markov's inequality that

$$\begin{aligned} \Pr\left(V_i \text{ is residual} \mid V_i \text{ is non-simple}\right) &\leq \Pr\left(|N_{G_R}(V_i)| \geq \frac{1}{2} \cdot n^{\delta/2}\right) \\ &\leq \frac{2 \cdot 80 \log^4 n}{n^{\delta/2}} \leq \frac{\log^5 n}{n^{\delta/2}}. \end{aligned} \tag{7.1}$$

Let X_i be the indicator random variable that a non-simple vertex group V_i is a residual vertex group, then $R = \sum_{i \in [\frac{n}{\alpha}] \setminus \mathcal{I}_s} X_i$ is the number of residual vertex groups. By Equation (7.1), we have the following:

$$\mathbb{E}[R] = \sum_{i \in [\frac{n}{\alpha}] \setminus \mathcal{I}_s} \Pr(X_i) \leq \sum_{i \in [\frac{n}{\alpha}]} \frac{\log^5 n}{n^{\delta/2}} = \frac{n \cdot \log^5 n}{\alpha \cdot n^{\delta/2}}.$$

Finally, since $\text{opt}(G) \geq \frac{n}{\alpha \cdot \log^2 n}$ (Assumption 7.6), a further application of Markov's inequality implies the result:

$$\Pr(|\mathcal{I}_r| > \text{opt}) \leq \Pr\left(R > \frac{n}{\alpha \cdot \log^2 n}\right) \leq \frac{n \cdot \log^5 n}{\alpha \cdot n^{\delta/2}} \cdot \frac{\alpha \log^2 n}{n} \leq n^{-\delta/4}.$$

Note that we can easily increase the success probability by running the algorithm in parallel $40/\delta$ times and detecting failures when the number of residual groups is more than $n/\alpha \cdot \log^2 n$. Then, with probability at least $1 - n^{-10}$, one of the runs will succeed. This only increases the space of the algorithm by a constant factor since $40/\delta = \Theta(1)$. □

Clean Vertex Groups. Let $\mathcal{I}_c$ be the index set of the clean vertex groups and let $\mathcal{I}_c^+$ be the ones added to the solution, which also corresponds to the group-level vertex cover V_C' in Algorithm 5.

Before analysing the group-level vertex cover, we note that the relevant counters are stored modulo c. This means that if the number of edges between clean vertex groups is some multiple of c, the corresponding counter would be 0 and the group-level vertex cover would be incorrect. Hence, we want the number of edges between clean vertex groups to be less than c with high probability.

Claim 7.12. *For all pairs of clean vertex groups V_i and V_j, with high probability,*

$$|N_G(V_i) \cap V_j| < c.$$

Proof. We prove a slightly generalised statement which implies what we need. We show that there are fewer than c edges of G_R between any clean vertex group V_i and any other vertex group V_j. This implies what we need since, by definition, all edges between clean vertex groups are in G_R.

Consider the random partitioning of V using an at least $(3 \cdot c)$-wise independent hash function (the algorithm uses $(10 \cdot \log n)$-wise independence). A residual neighbour of the clean vertex group $v \in N_{G_R}(V_i)$ uniformly belongs to any of the other vertex groups. Since there are $\frac{n}{\alpha} - 1$ of these (including V_j, but not including V_i), the probability that $v \in V_j$ is at most $\frac{2\alpha}{n}$.

Now, since clean vertex groups are non-residual, $|N_{G_R}(V_i)| \leq n^{\delta/2}$, and for a fixed V_i and V_j, we have that

$$\Pr\left(|N_{G_R}(V_i) \cap V_j| \geq c\right) \leq \binom{n^{\delta/2}}{c} \cdot \left(\frac{2\alpha}{n}\right)^c \leq \left(\frac{2\alpha}{n^{1-\delta/2}}\right)^{15/\delta} \leq n^{-7}$$

where we have used $\alpha \leq n^{1-\delta}$ and $c = 15/\delta$ in the final inequalities. Then, the result holds with probability at least $1 - n^{-5}$ by a union bound over all pairs of vertex groups.

Note that for small α we will partition V into groups of size exactly α with a uniform random permutation due to concentration and space reasons (see Claim 7.15), but the above arguments also hold in this case. □

With Claim 7.12, we can assume that all the counters between clean vertex groups count exactly the number of edges with high probability, that is, the modulo has no effect on the correctness of the algorithm. Thus, the setting is now identical to that of Dark and Konrad's algorithm [DK20], and we follow a similar argument as they did to analyse the group-level vertex cover and the corresponding subset of clean vertex groups added.

Claim 7.13. *The number of clean vertex groups added $|\mathcal{I}_c^+|$ is at most $2 \cdot \text{opt}(G)$.*

Proof. Consider the subgraph $H = G[\cup_{i \in \mathcal{I}_c} V_i]$ induced by the clean vertex groups. Observe that since H is an induced subgraph of G, $\text{opt}(H) \leq \text{opt}(G)$. Then, since the vertex contractions to obtain the multi-graph G' from H cannot increase the size of its minimum vertex cover, we have that $\text{opt}(G') \leq \text{opt}(H)$. Finally, since we greedily compute the group-level vertex cover V'_C, it is a 2-approximation and we have that $|\mathcal{I}_c^+| = |V'_C| \leq 2 \cdot \text{opt}(G') \leq 2 \cdot \text{opt}(G)$. □

By combining the analysis of the simple, residual, and clean vertex groups, we prove the approximation factor of the algorithm.

Lemma 7.14. *Algorithm 5 returns a valid $\Theta(\alpha)$-approximation of a minimum vertex cover for any input graph G with $\text{opt} \geq \frac{n}{\alpha \cdot \log^2 n}$.*

Proof. We first show that the solution V_C is indeed a valid vertex cover, then we prove that it is a $\Theta(\alpha)$-approximation.

Validity. For the sake of finding a contradiction, let $e \in E$ be an edge which is not covered by V_C. Observe that any non-clean vertex group V_i is added to V_C; thus, all edges with at least one endpoint in any of these vertex groups are covered. So, we have that e must be in $G[\cup_{i \in \mathcal{I}_c} V_i]$, the subgraph induced by the clean vertex groups.

Let $i, j \in \mathcal{I}_c$ be such that e has endpoints in the clean vertex groups V_i and V_j, implying that there is an edge between their corresponding contracted vertices v_i and v_j in the multigraph G'. It follows that one of v_i or v_j must be in the computed group-level vertex cover V'_C, so all vertices of either V_i or V_j, including at least one endpoint of e, are added to V_C. However, this means that e is covered by V_C, a contradiction.

Approximation. Observe that the solution V_C is comprised of a (disjoint) union of all simple vertex groups, all residual vertex groups, and a subset of clean vertex groups. Recall that $\mathcal{I}_s$, $\mathcal{I}_r$ and $\mathcal{I}_c^+$ are the corresponding index sets of these groups.

By Claims 7.10, 7.11 and 7.13, we have that $|\mathcal{I}_s| + |\mathcal{I}_r| + |\mathcal{I}_c^+| \leq 5 \cdot \text{opt}$. Finally, since the size of each vertex group is at most 2α, we can bound the size of the solution as follows:

$$|V_C| = \sum_{i \in \mathcal{I}_s \cup \mathcal{I}_r \cup \mathcal{I}_c^+} |V_i| \leq 2\alpha \cdot (|\mathcal{I}_s| + |\mathcal{I}_r| + |\mathcal{I}_c^+|) \leq 10\alpha \cdot \text{opt}. \qquad \square$$

It remains to show that the algorithm can be implemented using $O(n^2/\alpha^2)$ bits of space. Algorithm 5 randomly partitions V, maintains several instances of $\mathcal{ALG}_{MS}$ (Lemma 7.5) and $\mathcal{ALG}_{NT}$ (Proposition 7.4), and stores various counters. To show the space usage of the algorithm, we first consider each of these components separately.

Claim 7.15. *The partitioning of V into $\frac{n}{\alpha}$ vertex groups of size in the range $[\alpha/2, 2\alpha]$ uses $O(n^2/\alpha^2)$ bits of space and succeeds with high probability.*

Proof. We show that for small $\alpha < \log^2 n$, i.e, when we have sufficient space, we can achieve this with a uniform random permutation, and for large $\alpha \geq \log^2 n$, we use a $(10 \cdot \log n)$-wise independent hash function.

Small α. For any $\alpha < \log^2 n$, we can randomly permute the vertices using $O(n \log n) = O(n^2/\alpha^2)$ random bits to create a uniform random partitioning of V into $\frac{n}{\alpha}$ groups of size α.

Large α. For any $\alpha \geq \log^2 n$, we can partition V using a $(10 \cdot \log n)$-wise independent hash function $h : [n] \to [\frac{n}{\alpha}]$ which uses $O(\log^2 n) = O(n^2/\alpha^2)$ bits by Proposition 7.2. We bound the size of the groups as follows: Consider any group V_j ($j \in [\frac{n}{\alpha}]$) and let X_i be the random variable that is 1 if vertex i is hashed to V_j, i.e., $h(i) = j$. Let $X = \sum_i X_i$ represent the number of vertices in group V_j. We have $\mathbb{E}[X] = n \cdot (\alpha/n) = \alpha$. Using Proposition 7.3 with $\varepsilon = 0.1$,

$$\Pr(|X - \mathbb{E}[X]| \geq \varepsilon \cdot \mathbb{E}[X]) \leq \exp(-5 \log n) \leq n^{-5}.$$

A union bound over all groups implies that with probability at least $1 - n^{-4}$, all groups have size between 0.9α and 1.1α. $\square$

Claim 7.16. *The instances of $\mathcal{ALG}_{MS}$ and $\mathcal{ALG}_{NT}$, and the counters use $O(n^2/\alpha^2)$ bits of space.*

Proof. We use one instance of $\mathcal{ALG}_{MS}$ (Lemma 7.5) which takes space $O(n^2/\alpha^2)$ bits. We use n/α instances of $\mathcal{ALG}_{NT}$ (Proposition 7.4) with parameters $a = n/\alpha$ and $b = n^{\delta/2}$ each of which take space $O(\frac{a}{b}\log^3 n) = O((n/\alpha)\cdot(\log^3 n/n^{\delta/2})) = o(n/\alpha)$ bits. This implies that the total space used by n/α instances is $O(n^2/\alpha^2)$ bits. We maintain counters modulo a constant $c = 15/\delta$ for the number of edges between every pair of vertex groups. Each takes $O(1)$ bits of space, and since there are $O(n^2/\alpha^2)$ many of these counters, this totals $O(n^2/\alpha^2)$ bits of space. We also maintain counters for the number of internal edges for each group which requires $O(\log n) = o(n/\alpha)$ bits of space each. Since there are $\frac{n}{\alpha}$ many groups, this totals $O(n^2/\alpha^2)$ bits of space. □

Hence, by Claims 7.15 and 7.16, we have shown that the components of Algorithm 5 use $O(n^2/\alpha^2)$ bits in total. We still, however, need to consider the format of the output. When α gets large enough, the space is only $o(n)$, whereas simply storing the output – the vertices of a solution – could require $\Theta(n)$ bits of space. We solve this by showing that we can implicitly store the solution when there is limited space.

Claim 7.17. *The output of Algorithm 5 can be maintained using $O(n^2/\alpha^2)$ bits of space.*

Proof. For $\alpha < \log^2 n$, we can maintain the vertices of the solution explicitly. For $\alpha \geq \log^2 n$, we rely on the hash function h used to partition V (see Claim 7.15). Recall that vertices are added to the solution at a group level, so we can simply maintain a bit vector of length $\frac{n}{\alpha}$ representing the groups added to the solution. Then, the output consists of h and the bit vector which is sufficient for checking if a vertex belongs to the solution and uses $O(\log^2 n + \frac{n}{\alpha}) = O(n^2/\alpha^2)$ bits of space. □

We have now shown that Algorithm 5 can be implemented using $O(n^2/\alpha^2)$ bits of space. Therefore, combined with Lemma 7.14 and Assumption 7.6, we have proven our main result, Theorem 7.1.

7.5 Conclusion

In this chapter, we have resolved the space complexity of αMVC for the full range of α. We have provided a randomised algorithm which asymptotically matches the lower bound [DK20] up to constant factors, showing that $\Theta(n^2/\alpha^2)$ is necessary and sufficient for this problem.

Our work continues the direction set by the results on connectivity [AGM12a, NY19] and matchings [DK20, AS22]; we resolve the space complexity (up to constant factors) of another problem in the insertion-deletion streaming setting. However, other problems still have a

polylogarithmic gap. In particular, can we achieve space optimality up to constant factors for insertion-deletion streaming problems such as dominating set [KK22] and spectral sparsification [KMM+20]?

More interestingly (at least to us), is considering the deterministic version of αMVC. The previous best algorithm for αMVC was a deterministic one using $O(n^2/\alpha^2 \cdot \log \alpha)$ bits of space [DK20] and we have shown that it is possible to remove the logarithmic overhead using randomness. Is it possible, however, to remove this logarithmic overhead using deterministic techniques or otherwise prove a deterministic lower bound which shows that it is necessary?

Considering again the deterministic algorithm in [DK20], the counters that introduce the logarithmic overhead are crucial for determining whether a set (of edges) is empty or not at the end of the stream, which could possibly get very large during the stream. Our randomised algorithm tackles this by guaranteeing that the size of the sets are constant at the end of the stream. Whether a deterministic strategy could obtain a similar guarantee is the goal of obtaining a better deterministic algorithm for αMVC.

On the other hand, if we assume that the above is not possible, then the goal would be to develop a streaming strategy that can differentiate when a set has exactly 0 elements or up to α^2 many using constant space. This relates very closely to the equality testing problem in such that, for some split of the stream into two parts, we essentially need to be able to test whether the current (possibly negative) count of elements in a set w.r.t. the first part of the stream is equal to the negative of the count of elements in the same set w.r.t. the second part of the stream. It is well-known that, for deterministic protocols, the optimal protocol is to communicate the entire count, which indicates that there is no hope to do better. Hence, the goal in this case would be to exploit this fact and prove an improved deterministic lower bound.

Acknowledgements

We are grateful to Sepehr Assadi and Christian Konrad for many helpful discussions. We also appreciate the valuable comments from our APPROX 2022 reviewers.

Bibliography

[AB21] Sepehr Assadi and Soheil Behnezhad. Beating two-thirds for random-order streaming matching. 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 19:1–19:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. Cited on pages 2, 3, 50, 89, 111, and 117.

[ABB+19] Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab S. Mirrokni, and Cliff Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1616–1635. SIAM, 2019. Cited on pages 3 and 138.

[ABKL23] Sepehr Assadi, Soheil Behnezhad, Sanjeev Khanna, and Huan Li. On regularity lemma and barriers in streaming and dynamic matching. 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 131–144. ACM, 2023. Cited on page 111.

[ABR24] Amir Azarmehr, Soheil Behnezhad, and Mohammad Roghani. Fully dynamic matching: -approximation in polylog update time. 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 3040–3061. SIAM, 2024. Cited on pages 3, 5, 6, and 111.

[ACG+15] Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2237–2246. JMLR.org, 2015. Cited on pages 2, 3, 19, 52, 134, and 140.

[ACK19a] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Polynomial pass lower bounds for graph streaming algorithms. In Moses Charikar and Edith Cohen, editors, *Pro-*

ceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019, pages 265–276. ACM, 2019. Cited on page 138.

[ACK19b] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for (Δ + 1) vertex coloring. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 767–786. SIAM, 2019. Cited on pages 2, 3, 50, and 136.

[ACK21] Sepehr Assadi, Deeparnab Chakrabarty, and Sanjeev Khanna. Graph Connectivity and Single Element Recovery via Linear and OR Queries. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms (ESA 2021)*, volume 204 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Cited on pages 27 and 59.

[AD21] Sepehr Assadi and Aditi Dudeja. A simple semi-streaming algorithm for global minimum cuts. In Hung Viet Le and Valerie King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 172–180. SIAM, 2021. Cited on pages 2 and 3.

[ADKN23] Cezar-Mihail Alexandru, Pavel Dvorák, Christian Konrad, and Kheeran K. Naidu. Improved weighted matching in the sliding window model. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 6:1–6:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. Cited on pages 2 and 111.

[AG09] Kook Jin Ahn and Sudipto Guha. Graph sparsification in the semi-streaming model. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II*, volume 5556 of *Lecture Notes in Computer Science*, pages 328–338. Springer, 2009. Cited on page 3.

[AG11] Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Automata, Languages and Programming - 38th*

International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II, volume 6756 of *Lecture Notes in Computer Science*, pages 526–538. Springer, 2011. Cited on pages 3 and 6.

[AG13] Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.*, 222:59–79, 2013. Cited on page 57.

[AG15] Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. In Guy E. Blelloch and Kunal Agrawal, editors, *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*, pages 202–211. ACM, 2015. Cited on pages 3 and 6.

[AGM12a] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 459–467. SIAM, 2012. Cited on pages 3, 10, 136, and 148.

[AGM12b] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini, editors, *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 5–14. ACM, 2012. Cited on pages 3, 10, and 136.

[AJJ+22] Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 627–669. SIAM, 2022. Cited on pages 3, 6, 10, 57, 111, and 138.

[AKL16] Sepehr Assadi, Sanjeev Khanna, and Yang Li. Tight bounds for single-pass streaming complexity of the set cover problem. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 698–711. ACM, 2016. Cited on page 119.

[AKLY16] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model.

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 1345–1364. SIAM, 2016. Cited on pages 3, 9, 10, 11, 89, 136, and 137.

[AKM22] Sepehr Assadi, Pankaj Kumar, and Parth Mittal. Brooks' theorem in graph streams: a single-pass semi-streaming algorithm for Δ-coloring. 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 234–247. ACM, 2022. Cited on page 3.

[AKNS24] Sepehr Assadi, Christian Konrad, Kheeran K. Naidu, and Janani Sundaresan. O(log log n) passes is optimal for semi-streaming maximal independent set. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, British Columbia, Canada, June 24-28, 2024*. ACM, 2024. Cited on pages 2, 3, 9, 19, 50, 133, and 134.

[AKSY20] Sepehr Assadi, Gillat Kol, Raghuvansh R. Saxena, and Huacheng Yu. Multi-pass graph streaming lower bounds for cycle counting, max-cut, matching size, and other problems. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 354–364. IEEE, 2020. Cited on page 138.

[AKZ24] Sepehr Assadi, Gillat Kol, and Zhijun Zhang. Optimal multi-pass lower bounds for MST in dynamic streams. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, British Columbia, Canada, June 24-28, 2024*. ACM, 2024. Cited on page 3.

[ALT21] Sepehr Assadi, S. Cliff Liu, and Robert E. Tarjan. An auction algorithm for bipartite matching in streaming and massively parallel computation models. In Hung Viet Le and Valerie King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 165–171. SIAM, 2021. Cited on pages 3, 6, 10, 57, 58, and 111.

[AMM22] Raghavendra Addanki, Andrew McGregor, and Cameron Musco. Non-adaptive edge counting and sampling via bipartite independent set queries. *CoRR*, abs/2207.02817, 2022. Cited on pages 27, 59, and 60.

[AMS96] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadel-*

phia, Pennsylvania, USA, May 22-24, 1996, pages 20–29. ACM, 1996. Cited on page 1.

[AN21] Sepehr Assadi and Vishvajeet N. Graph streaming lower bounds for parameter estimation and property testing via a streaming XOR lemma. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 612–625. ACM, 2021. Cited on pages 50 and 117.

[AR20] Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 342–353. IEEE, 2020. Cited on pages 3, 4, 7, 9, 43, 50, 111, 117, and 118.

[AS22] Sepehr Assadi and Vihan Shah. An asymptotically optimal algorithm for maximum matching in dynamic streams. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 9:1–9:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. Cited on pages 3, 7, 10, 11, 20, 52, 53, 89, 111, 136, 137, 138, 139, 140, 148, 171, 173, and 175.

[AS23a] Sepehr Assadi and Janani Sundaresan. Hidden permutations to the rescue: Multi-pass streaming lower bounds for approximate matchings. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 909–932. IEEE, 2023. Cited on pages 3, 7, 9, 19, 43, 50, 118, and 134.

[AS23b] Sepehr Assadi and Janani Sundaresan. (noisy) gap cycle counting strikes back: Random order streaming lower bounds for connected components and beyond. 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 183–195. ACM, 2023. Cited on page 3.

[Ass22] Sepehr Assadi. A two-pass (conditional) lower bound for semi-streaming maximum matching. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 708–742. SIAM, 2022. Cited on pages x, 3, 7, 9, 17, 18, 43, 50, 88, 89, 111, 117, 118, and 138.

[Ass23] Sepehr Assadi. Recent advances in multi-pass graph streaming lower bounds. *SIGACT News*, 54(3):48–75, 2023. Cited on page 43.

[Ass24] Sepehr Assadi. A simple (1 - ϵ)-approximation semi-streaming algorithm for maximum (weighted) matching. In Merav Parter and Seth Pettie, editors, *2024 Symposium on Simplicity in Algorithms, SOSA 2024, Alexandria, VA, USA, January 8-10, 2024*, pages 337–354. SIAM, 2024. Cited on pages 3 and 6.

[BdBM21] Leyla Biabani, Mark de Berg, and Morteza Monemizadeh. Maximum-weight matching in sliding windows and beyond. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd International Symposium on Algorithms and Computation, ISAAC 2021, December 6-8, 2021, Fukuoka, Japan*, volume 212 of *LIPIcs*, pages 73:1–73:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. Cited on pages 111 and 138.

[Beh21] Soheil Behnezhad. Time-optimal sublinear algorithms for matching and vertex cover. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 873–884. IEEE, 2021. Cited on pages 27 and 59.

[Beh23] Soheil Behnezhad. *Dynamic Algorithms for Maximum Matching Size*, pages 129–162. Society for Industrial and Applied Mathematics, 2023. Cited on pages 5 and 17.

[Ber20] Aaron Bernstein. Improved bounds for matching in random-order streams. 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 12:1–12:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. Cited on pages 2, 3, 89, 111, and 138.

[BHH19] Soheil Behnezhad, MohammadTaghi Hajiaghayi, and David G. Harris. Exponentially faster massively parallel maximal matching. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1637–1649. IEEE Computer Society, 2019. Cited on page 27.

[BHR+20] Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge estimation with independent set oracles. *ACM Trans. Algorithms*, 16(4):52:1–52:27, 2020. Cited on pages 27 and 59.

[BJKS02] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002,*

Vancouver, BC, Canada, Proceedings, pages 209–218. IEEE Computer Society, 2002. Cited on pages 25 and 119.

[BKSV14] Vladimir Braverman, Jonathan Katzman, Charles Seidell, and Gregory Vorsanger. An optimal algorithm for large frequency moments using o(n^(1-2/k)) bits. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, volume 28 of *LIPIcs*, pages 531–544. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014. Cited on page 138.

[BKSW23] Sayan Bhattacharya, Peter Kiss, Thatchaphol Saranurak, and David Wajc. Dynamic matching with better-than-2 approximation in polylogarithmic update time. 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 100–128. SIAM, 2023. Cited on pages 3, 4, 5, 6, 7, 8, 14, 15, 16, 17, 18, 30, 36, 88, and 133.

[BLS+18] Vladimir Braverman, Zaoxing Liu, Tejasvam Singh, N. V. Vinodchandran, and Lin F. Yang. New bounds for the CLIQUE-GAP problem using graph decomposition theory. *Algorithmica*, 80(2):652–667, 2018. Cited on page 2.

[Bra17] Mark Braverman. Interactive information complexity. *SIAM Rev.*, 59(4):803–846, 2017. Cited on page 112.

[CCE+16] Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. 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 1326–1344. SIAM, 2016. Cited on pages 3, 10, 11, 89, 107, 136, 137, and 142.

[CCHM15] Rajesh Hemant Chitnis, Graham Cormode, Mohammad Taghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1234–1251. SIAM, 2015. Cited on page 3.

[CCKM13] Amit Chakrabarti, Graham Cormode, Ranganath Kondapally, and Andrew McGregor. Information cost tradeoffs for augmented index and streaming language recognition. *SIAM J. Comput.*, 42(1):61–83, 2013. Cited on page 113.

[CDK19] Graham Cormode, Jacques Dark, and Christian Konrad. Independent sets in vertex-arrival streams. 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 45:1–45:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. Cited on pages 2 and 50.

[CEK+15] Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. One trillion edges: graph processing at facebook-scale. *Proc. VLDB Endow.*, 8(12):1804–1815, aug 2015. Cited on page 1.

[CK11] Amit Chakrabarti and Ranganath Kondapally. Everywhere-tight information cost tradeoffs for augmented index. In Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*, volume 6845 of *Lecture Notes in Computer Science*, pages 448–459. Springer, 2011. Cited on pages 113 and 169.

[CKP+21] Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, Zhao Song, and Huacheng Yu. Almost optimal super-constant-pass streaming lower bounds for reachability. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 570–583. ACM, 2021. Cited on pages 3, 4, 7, 9, 43, 50, 111, 117, and 118.

[CLRS22] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms, fourth edition.* MIT Press, 2022. Cited on page 1.

[CMS13] Michael S. Crouch, Andrew McGregor, and Daniel M. Stubbs. Dynamic graphs in the sliding-window model. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, volume 8125 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 2013. Cited on pages 2, 3, 20, and 138.

[CS14] Michael S. Crouch and Daniel M. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, volume 28 of

LIPIcs, pages 96–104. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014. Cited on page 138.

[CSWY01] Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, and Andrew Chi-Chih Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 270–278. IEEE Computer Society, 2001. Cited on pages 25 and 119.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006. Cited on page 119.

[DG04] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In Eric A. Brewer and Peter Chen, editors, *6th Symposium on Operating System Design and Implementation (OSDI 2004), San Francisco, California, USA, December 6-8, 2004*, pages 137–150. USENIX Association, 2004. Cited on page 1.

[DK20] Jacques Dark and Christian Konrad. Optimal lower bounds for matching and vertex cover in dynamic graph streams. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. Cited on pages 3, 7, 9, 10, 11, 20, 50, 52, 53, 54, 89, 111, 136, 137, 141, 146, 148, and 149.

[DP09] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009. Cited on page 171.

[EHM16] Hossein Esfandiari, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Finding large matchings in semi-streaming. In Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 608–614. IEEE Computer Society, 2016. Cited on pages 3, 4, 5, 6, 7, 8, 10, 14, 15, 16, 18, 30, 36, 38, 58, 85, 86, 97, 101, and 111.

[EKMS12] Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite matching in the semi-streaming model. *Algorithmica*, 63(1-2):490–508, 2012. Cited on pages 6, 10, and 57.

[EKS09] Sebastian Eggert, Lasse Kliemann, and Anand Srivastav. Bipartite graph matchings in the semi-streaming model. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 492–503. Springer, 2009. Cited on page 3.

[Fei06] Uriel Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM J. Comput.*, 35(4):964–984, 2006. Cited on pages 27 and 59.

[FHM+20] Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi. Approximate maximum matching in random streams. 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 1773–1785. SIAM, 2020. Cited on pages 3, 89, and 111.

[FHS17] Jacob Fox, Hao Huang, and Benny Sudakov. On graphs decomposable into induced matchings of linear sizes. *Bulletin of the London Mathematical Society*, 49(1):45–57, 2017. Cited on pages 18, 24, 50, 88, and 112.

[FKM+04] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, volume 3142 of *Lecture Notes in Computer Science*, pages 531–543. Springer, 2004. Cited on pages 2, 3, 5, 6, 8, 9, 10, 12, 15, 18, 30, 56, 58, 111, and 136.

[FKM+05] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the streaming model: the value of space. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 745–754. SIAM, 2005. Cited on pages 2 and 10.

[FLN+02] Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 474–483. ACM, 2002. Cited on pages 18, 24, 41, 50, 88, 112, and 125.

[FLN14] Jittat Fakcharoenphol, Bundit Laekhanukit, and Danupon Nanongkai. Faster algorithms for semi-matching problems. *ACM Trans. Algorithms*, 10(3), May 2014. Cited on page 86.

[FMU22] Manuela Fischer, Slobodan Mitrovic, and Jara Uitto. Deterministic $(1+\epsilon)$-approximate maximum matching with poly$(1/\epsilon)$ passes in the semi-streaming model 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 248–260. ACM, 2022. Cited on pages 3 and 6.

[FNSZ23] Moran Feldman, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. The one-way communication complexity of submodular maximization with applications to streaming and robustness. *J. ACM*, 70(4):24:1–24:52, 2023. Cited on page 46.

[FS22] Moran Feldman and Ariel Szarf. Maximum matching sans maximal matching: A new approach for finding maximum matchings in the data stream model. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPIcs*, pages 33:1–33:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. Cited on pages 3, 4, 5, 6, 8, 10, 14, 15, 18, 30, 58, and 83.

[GGK+18] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrovic, and Ronitt Rubinfeld. Improved massively parallel computation algorithms for mis, matching, and vertex cover. In Calvin Newport and Idit Keidar, editors, *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 129–138. ACM, 2018. Cited on page 2.

[GKK12] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485. SIAM, 2012. Cited on pages x, xiv, 3, 7, 9, 17, 18, 24, 40, 41, 42, 43, 44, 45, 49, 50, 85, 86, 91, 92, 95, 111, 112, 117, 125, and 133.

[GKMS19] Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 491–500. ACM, 2019. Cited on pages 3, 52, and 140.

[GO13] Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 287–298. IEEE Computer Society, 2013. Cited on pages 3, 4, 7, and 43.

[GO16] Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. *Algorithmica*, 76(3):654–683, 2016. Cited on page 118.

[GR08] Oded Goldreich and Dana Ron. Approximating average parameters of graphs. *Random Struct. Algorithms*, 32(4):473–493, 2008. Cited on pages 27 and 59.

[GU19] Mohsen Ghaffari and Jara Uitto. Sparsifying distributed algorithms with ramifications in massively parallel computation and centralized local computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, page 1636–1653, USA, 2019. Society for Industrial and Applied Mathematics. Cited on page 27.

[HK73] 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. Cited on page 4.

[HPT+19] Zhiyi Huang, Binghui Peng, Zhihao Gavin Tang, Runzhou Tao, Xiaowei Wu, and Yuhao Zhang. Tight competitive ratios of classic matching algorithms in the fully online model. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2875–2886. SIAM, 2019. Cited on pages 17 and 88.

[HSSW12] Magnús M. Halldórsson, Xiaoming Sun, Mario Szegedy, and Chengu Wang. Streaming and communication complexity of clique approximation. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, volume 7391 of *Lecture Notes in Computer Science*, pages 449–460. Springer, 2012. Cited on page 2.

[IPW11] Piotr Indyk, Eric Price, and David P. Woodruff. On the power of adaptivity in sparse recovery. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 285–294. IEEE Computer Society, 2011. Cited on page 138.

[JN14] Rahul Jain and Ashwin Nayak. The space complexity of recognizing well-parenthesized expressions in the streaming model: The index function revisited. *IEEE Trans. Inf. Theory*, 60(10):6646–6668, 2014. Cited on page 113.

[JRS09] Rahul Jain, Jaikumar Radhakrishnan, and Pranab Sen. A property of quantum relative entropy with an application to privacy in quantum communication. *J. ACM*, 56(6):33:1–33:32, 2009. Cited on pages 19, 46, 47, 112, and 113.

[JST11] Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 49–58. ACM, 2011. Cited on pages 10, 52, and 136.

[Kap13] Michael Kapralov. Better bounds for matchings in the streaming model. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1679–1697. SIAM, 2013. Cited on pages 3, 7, 18, 43, 85, 111, and 133.

[Kap21] Michael Kapralov. Space lower bounds for approximating maximum matching in the edge arrival model. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1874–1893. SIAM, 2021. Cited on pages 3, 7, 17, 18, 43, 50, 56, 85, 88, 111, and 133.

[KK20] Lidiya Khalidah binti Khalil and Christian Konrad. Constructing large matchings via query access to a maximal matching oracle. In Nitin Saxena and Sunil Simon, editors, *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020, December 14-18, 2020, BITS Pilani, K K Birla Goa Campus, Goa, India (Virtual Conference)*, volume 182 of *LIPIcs*, pages 26:1–26:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. Cited on pages xiv, 7, 8, 14, 27, 32, 56, 57, 58, 59, 60, 70, and 111.

[KK22] Sanjeev Khanna and Christian Konrad. Optimal bounds for dominating set in graph streams. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 93:1–93:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. Cited on pages 10, 136, 137, and 149.

[KLM+14] Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 561–570. IEEE Computer Society, 2014. Cited on pages 3 and 136.

[KMM12] Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In Anupam Gupta, Klaus Jansen, José D. P. Rolim, and Rocco A. Servedio, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, volume 7408 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2012. Cited on pages 3, 4, 5, 6, 8, 12, 15, 16, 18, 30, 36, 37, 38, 39, 58, 85, 89, 98, 111, and 138.

[KMM+20] Michael Kapralov, Aida Mousavifar, Cameron Musco, Christopher Musco, Navid Nouri, Aaron Sidford, and Jakab Tardos. Fast and space efficient spectral sparsification in dynamic streams. 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 1814–1833. SIAM, 2020. Cited on pages 136, 137, and 149.

[KN96] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1996. Cited on page 112.

[KN21] Christian Konrad and Kheeran K. Naidu. On two-pass streaming algorithms for maximum bipartite matching. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 19:1–19:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. Cited on pages 3, 4, 5, 6, 7, 8, 10, 11, 14, 16, 21, 30, 35, 43, 84, 88, 89, 111, 133, and 138.

[KN24] Christian Konrad and Kheeran K. Naidu. An unconditional lower bound for two-pass streaming algorithms for maximum matching approximation. 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 2881–2899. SIAM, 2024. Cited on pages 3, 7, 9, 11, 17, 21, 43, 89, 110, and 134.

[KNP+17] Michael Kapralov, Jelani Nelson, Jakub Pachocki, Zhengyu Wang, David P. Woodruff, and Mobin Yahyazadeh. Optimal lower bounds for universal relation, and for samplers and finding duplicates in streams. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 475–486. IEEE Computer Society, 2017. Cited on pages 10, 52, and 136.

[KNS23] Christian Konrad, Kheeran K. Naidu, and Arun Steward. Maximum matching via maximal matching queries. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar,

and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 41:1–41:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. Cited on pages 3, 4, 5, 6, 7, 8, 10, 11, 12, 21, 27, 29, 55, and 111.

[Kon15] Christian Konrad. Maximum matching in turnstile streams. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 840–852. Springer, 2015. Cited on pages 3, 10, 89, and 136.

[Kon18a] Christian Konrad. MIS in the congested clique model in o(log log Δ) rounds. *CoRR*, abs/1802.07647, 2018. Cited on pages 2, 52, and 140.

[Kon18b] Christian Konrad. A simple augmentation method for matchings with applications to streaming algorithms. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 74:1–74:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. Cited on pages 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 15, 16, 17, 18, 30, 36, 38, 39, 57, 58, 85, 89, 97, 98, 100, 101, 105, 106, 111, 133, and 138.

[Kon21] Christian Konrad. Frequent elements with witnesses in data streams. In Leonid Libkin, Reinhard Pichler, and Paolo Guagliardo, editors, *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, pages 83–95. ACM, 2021. Cited on pages 10 and 136.

[KOT24] Christian Konrad, Conor O'Sullivan, and Victor Traistaru. Graph reconstruction via MIS queries. *CoRR*, abs/2401.05845, 2024. Cited on page 27.

[KR16] Christian Konrad and Adi Rosén. Approximating semi-matchings in streaming and in two-party communication. *ACM Trans. Algorithms*, 12(3), April 2016. Cited on page 86.

[KSV10] Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 938–948. SIAM, 2010. Cited on pages 56 and 57.

[KT17] Sagar Kale and Sumedh Tirodkar. Maximum matching in two, three, and a few more passes over graph streams. In Klaus Jansen, José D. P. Rolim, David

Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, volume 81 of *LIPIcs*, pages 15:1–15:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. Cited on pages 3, 4, 5, 6, 7, 8, 10, 12, 14, 15, 16, 18, 29, 30, 36, 38, 58, 85, 86, 97, and 111.

[Kyp21] Stefanos Kypritidis. Facebook network analysis. In NetworkX at https://networkx.org/nx-guides/content/exploratory_notebooks/facebook_notebook.html. See also https://github.com/networkx/nx-guides/tree/main/content/exploratory_notebooks [Accessed: 15 March 2024], 2021. Cited on page 1.

[Lin87] Nathan Linial. Distributive graph algorithms-global solutions from local data. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 331–335. IEEE Computer Society, 1987. Cited on page 1.

[LP09] L. Lovász and M.D. Plummer. *Matching Theory*. AMS Chelsea Publishing Series. AMS Chelsea Pub., 2009. Cited on page 23.

[LSZ+23] S. Cliff Liu, Zhao Song, Hengjie Zhang, Lichen Zhang, and Tianyi Zhou. Space-efficient interior point method, with applications to linear programming and maximum weight bipartite matching. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 88:1–88:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. Cited on page 3.

[McG05] Andrew McGregor. Finding graph matchings in data streams. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, volume 3624 of *Lecture Notes in Computer Science*, pages 170–181. Springer, 2005. Cited on pages 3 and 6.

[McG14] Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Rec.*, 43(1):9–20, 2014. Cited on pages 2, 25, 56, 111, and 138.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. Cited on page 139.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. Cited on pages 90 and 105.

[Mut05] S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2), 2005. Cited on pages 2, 25, 111, and 138.

[Nal21] Sai Krishna Chaitanya Nalam Venkata Subrahmanya. Vertex cover in the sliding window model. In Rutgers Library at `https://doi.org/doi:10.7282/t3-xyr3-1446`, 2021. Cited on page 20.

[NS22] Kheeran K. Naidu and Vihan Shah. Space optimal vertex cover in dynamic streams. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPIcs*, pages 53:1–53:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. Cited on pages 7, 10, 11, 19, 21, 51, and 135.

[NY19] Jelani Nelson and Huacheng Yu. Optimal lower bounds for distributed and streaming spanning forest computation. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1844–1860. SIAM, 2019. Cited on pages 3, 10, 136, and 148.

[OR10] Krzysztof Onak and Ronitt Rubinfeld. Maintaining a large matching and a small vertex cover. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 457–464, New York, NY, USA, 2010. Association for Computing Machinery. Cited on page 5.

[PW13] Eric Price and David P. Woodruff. Lower bounds for adaptive sparse recovery. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 652–663. SIAM, 2013. Cited on page 138.

[RS78] Imre Z Ruzsa and Endre Szemerédi. Triple systems with no six points carrying three triangles. *Combinatorics (Keszthely, 1976), Coll. Math. Soc. J. Bolyai*, 18:939–945, 1978. Cited on pages 7 and 40.

[RSW18] Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing exact minimum cuts without knowing the graph. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-*

14, 2018, Cambridge, MA, USA, volume 94 of *LIPIcs*, pages 39:1–39:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. Cited on page 2.

[SSS93] Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-hoeffding bounds for applications with limited independence. In Vijaya Ramachandran, editor, *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA*, pages 331–340. ACM/SIAM, 1993. Cited on page 139.

[SW15] Xiaoming Sun and David P. Woodruff. Tight bounds for graph problems in insertion streams. In Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, volume 40 of *LIPIcs*, pages 435–448. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. Cited on pages 2, 3, 10, 111, and 136.

[Tay23] Petroc Taylor. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025. In Statista at https://www.statista.com/statistics/871513/worldwide-data-created/ [Accessed: 12 March 2024], 2023. Cited on page 1.

[Tir18] Sumedh Tirodkar. Deterministic algorithms for maximum matching on general graphs in the semi-streaming model. In Sumit Ganguly and Paritosh K. Pandya, editors, *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India*, volume 122 of *LIPIcs*, pages 39:1–39:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. Cited on pages 3 and 6.

[Wil91] David Williams. *Probability with Martingales*. Cambridge mathematical textbooks. Cambridge University Press, 1991. Cited on page 90.

[WZ12] David P. Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 941–960. ACM, 2012. Cited on page 138.

[Yao77] Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Symposium on Foundations of Computer Science (FOCS)*, pages pp. 222–227. IEEE Computer Society, 1977. Cited on page 91.

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In Michael J. Fischer, Richard A. DeMillo, Nancy A.

Lynch, Walter A. Burkhard, and Alfred V. Aho, editors, *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 209–213. ACM, 1979. Cited on pages 25 and 119.

[Zel11] Mariano Zelke. Intractability of min- and max-cut in streaming graphs. *Inf. Process. Lett.*, 111(3):145–150, 2011. Cited on pages 2 and 3.

A Additional Proofs

A.1 Information Cost Trade-Off for Index

In this subsection, we show that the information cost trade-off result for Index can be obtained from the information cost trade-off result for *augmented index verification*, denoted AIV, by Chakrabarti and Kondapally [CK11].

The AIV_n problem, much like Index_n, is a two-party communication problem where the first party Alice holds an n-bit string $Y \in \{0,1\}^n$, and the second party Bob holds an index $J \in [n]$. In addition to this, however, Bob also holds the prefix $Y_{<J} = Y[1 : J-1]$ of Alice's input and a check-bit $C \in \{0,1\}$. The two parties communicate according to some protocol π_{AIV} until one of the parties outputs the bit $B_{\text{out}} := Y[J]$ XOR C. Note that we will only consider protocols, either for AIV or Index, where Bob outputs the solution since a protocol where Alice outputs the solution can be made into one where Bob does if Alice sends the output bit to Bob, which increases the information cost by at most one.

Let $\mathcal{D}_{\mathsf{AIV}}$ be the uniform distribution over the inputs (Y, J, C) and let $\mathcal{D}^0_{\mathsf{AIV}}$ be its corresponding 'easy' distribution $\mathcal{D}_{\mathsf{AIV}} \mid (B_{\text{out}} = 0)$ where the output bit B_{out} is always 0. Chakrabarti and Kondapally [CK11] prove the following information cost trade-off result:

Proposition A.1 (Information cost trade-off for AIV_n)**.** *Let $b \leq O(\log n)$ be an integer, and let $\delta < \frac{1}{2}$ be a positive constant. Any δ-error protocol π_{AIV} for AIV_n is such that:*

1. *Either $\mathsf{ICost}^A_{\mathcal{D}^0_{\mathsf{AIV}}}(\pi_{\mathsf{AIV}}) = \mathcal{I}_{\mathcal{D}^0_{\mathsf{AIV}}}(Y; \pi_{\mathsf{AIV}} \mid Y_{<J}, J, C, R_{\mathsf{AIV}}) \geq \frac{n}{2^{O(b)}}$, or*
2. *$\mathsf{ICost}^B_{\mathcal{D}^0_{\mathsf{AIV}}}(\pi_{\mathsf{AIV}}) = \mathcal{I}_{\mathcal{D}^0_{\mathsf{AIV}}}(J, C; \pi_{\mathsf{AIV}} \mid Y, R_{\mathsf{AIV}}) \geq b$,*

where the inputs are distributed according to the 'easy' distribution $\mathcal{D}^0_{\mathsf{AIV}}$.

We are now ready to prove the information cost trade-off result for Index.

Proposition 6.2 (Information cost trade-off for Index_n (restated))**.** Let $b \leq O(\log n)$ be an integer, and let $\delta < \frac{1}{2}$ be a positive constant. Any δ-error protocol π_{I} for Index_n is such that:

1. Either $\mathsf{ICost}^A_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) = \mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(Y; \pi_{\mathsf{I}} \mid J, R_{\mathsf{I}}) \geq \frac{n}{2^{O(b)}}$, or
2. $\mathsf{ICost}^B_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) = \mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(J; \pi_{\mathsf{I}} \mid Y, R_{\mathsf{I}}) \geq b$,

where $\mathcal{D}_{\mathsf{I}}$ denotes the uniform distribution.

Proof. Observe that π_{I} naturally yields a protocol π_{AIV} for AIV_n since, in AIV_n, Bob has more information than required for executing π_{I}, and, upon termination, Bob learns $Y[J]$ and can easily output $Y[J]$ XOR C. Hence, π_{AIV} errs only when π_{I} errs and is thus a δ-error protocol for AIV_n. Furthermore, the transcript of messages communicated and the public randomness used in π_{AIV} are exactly the same as in π_{I}.

Since $R_{\mathsf{AIV}} = R_{\mathsf{I}}$, $\pi_{\mathsf{AIV}} = \pi_{\mathsf{I}}$, and the inputs (Y, J) are distributed identically according to both $\mathcal{D}_{\mathsf{I}}$ and $\mathcal{D}_{\mathsf{AIV}}$, we have that

$$
\begin{aligned}
\mathsf{ICost}^{A}_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) &= \mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(Y; \pi_{\mathsf{I}} \mid J, R_{\mathsf{I}}) \\
&= \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(Y; \pi_{\mathsf{AIV}} \mid J, R_{\mathsf{AIV}}) \\
&\geq \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(Y; \pi_{\mathsf{AIV}} \mid Y_{<J}, J, R_{\mathsf{AIV}}) && \text{(since } Y_{<J} \text{ is contained in } Y\text{)} \\
&= \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(Y; \pi_{\mathsf{AIV}}, C, B_{\text{out}} \mid Y_{<J}, J, R_{\mathsf{AIV}}) - \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(Y; C, B_{\text{out}} \mid Y_{<J}, J, R_{\mathsf{AIV}}, \pi_{\mathsf{AIV}}) \\
& && \text{(by the chain rule (Fact 3))} \\
&\geq \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(Y; \pi_{\mathsf{AIV}}, C, B_{\text{out}} \mid Y_{<J}, J, R_{\mathsf{AIV}}) - 2 && \text{(since } H(C, B_{\text{out}}) \leq 2\text{)} \\
&= \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(Y; C, B_{\text{out}} \mid Y_{<J}, J, R_{\mathsf{AIV}}) + \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(Y; \pi_{\mathsf{AIV}} \mid Y_{<J}, J, C, B_{\text{out}}, R_{\mathsf{AIV}}) - 2 \\
& && \text{(by the chain rule (Fact 3))} \\
&\geq \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(Y; \pi_{\mathsf{AIV}} \mid Y_{<J}, J, C, B_{\text{out}}, R_{\mathsf{AIV}}) - 2 \\
&\geq \frac{1}{2} \cdot \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(Y; \pi_{\mathsf{AIV}} \mid Y_{<J}, J, C, R_{\mathsf{AIV}}, B_{\text{out}} = 0) - 2 && \text{(by Fact 1 since } B_{\text{out}} \text{ is uniform)} \\
&= \frac{1}{2} \cdot \mathcal{I}_{\mathcal{D}^0_{\mathsf{AIV}}}(Y; \pi_{\mathsf{AIV}} \mid Y_{<J}, J, C, R_{\mathsf{AIV}}) - 2 && \text{(since } \mathcal{D}^0_{\mathsf{AIV}} = \mathcal{D}_{\mathsf{AIV}} \mid (B_{\text{out}} = 0)\text{)} \\
&= \frac{1}{2} \cdot \mathsf{ICost}^{A}_{\mathcal{D}^0_{\mathsf{AIV}}}(\pi_{\mathsf{AIV}}) - 2\,. && \text{(A.1)}
\end{aligned}
$$

By a similar string of arguments, we further have that

$$
\begin{aligned}
\mathsf{ICost}^{B}_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) &= \mathcal{I}_{\mathcal{D}_{\mathsf{I}}}(J; \pi_{\mathsf{I}} \mid Y, R_{\mathsf{I}}) \\
&= \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(J; \pi_{\mathsf{AIV}} \mid Y, R_{\mathsf{AIV}}) \\
&= \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(J, C, B_{\text{out}}; \pi_{\mathsf{AIV}} \mid Y, R_{\mathsf{AIV}}) - \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(C, B_{\text{out}}; \pi_{\mathsf{AIV}} \mid Y, J, R_{\mathsf{AIV}},) \\
& && \text{(by the chain rule (Fact 3))} \\
&\geq \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(J, C, B_{\text{out}}; \pi_{\mathsf{AIV}} \mid Y, R_{\mathsf{AIV}}) - 2 && \text{(since } H(C, B_{\text{out}}) \leq 2\text{)} \\
&\geq \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(B_{\text{out}}; \pi_{\mathsf{AIV}} \mid Y, R_{\mathsf{AIV}}) + \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(J, C; \pi_{\mathsf{AIV}} \mid Y, B_{\text{out}}, R_{\mathsf{AIV}}) - 2 \\
& && \text{(by the chain rule (Fact 3))} \\
&\geq \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(J, C; \pi_{\mathsf{AIV}} \mid Y, B_{\text{out}}, R_{\mathsf{AIV}}) - 2 \\
&\geq \frac{1}{2} \cdot \mathcal{I}_{\mathcal{D}_{\mathsf{AIV}}}(J, C; \pi_{\mathsf{AIV}} \mid Y, R_{\mathsf{AIV}}, B_{\text{out}} = 0) - 2 && \text{(by Fact 1 since } B_{\text{out}} \text{ is uniform)} \\
&= \frac{1}{2} \cdot \mathcal{I}_{\mathcal{D}^0_{\mathsf{AIV}}}(J, C; \pi_{\mathsf{AIV}} \mid Y, R_{\mathsf{AIV}}) - 2 && \text{(since } \mathcal{D}^0_{\mathsf{AIV}} = \mathcal{D}_{\mathsf{AIV}} \mid (B_{\text{out}} = 0)\text{)}
\end{aligned}
$$

$$= \frac{1}{2} \cdot \mathsf{ICost}^{B}_{\mathcal{D}^{0}_{\mathsf{AIV}}}(\pi_{\mathsf{AIV}}) - 2 \ . \tag{A.2}$$

Now, suppose that $\mathsf{ICost}^{B}_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) \leq b$. By Inequality A.2, we have that $\mathsf{ICost}^{B}_{\mathcal{D}^{0}_{\mathsf{AIV}}}(\pi_{\mathsf{AIV}}) \leq 2b+4$, which, by Proposition A.1, implies that $\mathsf{ICost}^{B}_{\mathcal{D}^{0}_{\mathsf{AIV}}}(\pi_{\mathsf{AIV}}) \geq \frac{n}{2^{O(b)}}$. Finally, by Inequality A.1, we have that $\mathsf{ICost}^{B}_{\mathcal{D}_{\mathsf{I}}}(\pi_{\mathsf{I}}) \geq \frac{n}{2 \cdot 2^{O(b)}} - 2 = \frac{n}{2^{O(b)}}$. □

A.2 An altered Match-or-Sparsify

In this section, we prove that an altered version of Assadi and Shah's $\mathsf{Match\text{-}or\text{-}Sparsify}_n$ implies Lemma 7.5. Informally speaking, given a graph G, this lemma gives an algorithm that either finds a large matching in G or identifies a sparse induced subgraph of G.

The analysis we give follows the analysis of Assadi and Shah [AS22], and we give the full argument for completeness. We will also highlight the key alterations that we make. Before beginning, we give some required probabilistic and sketching tools in Appendix A.2.1 and Appendix A.2.2, respectively.

A.2.1 Probabilistic Tools

Proposition A.2 (Chernoff bound; c.f. [DP09])**.** *Suppose $X_1, \ldots, X_m$ are m independent random variables with range $[0,1]$ each. Let $X := \sum_{i=1}^{m} X_i$ and $\mu_L \leq \mathbb{E}[X] \leq \mu_H$. Then, for any $\varepsilon > 0$,*

$$\Pr\left(X > (1+\varepsilon) \cdot \mu_H\right) \leq \exp\left(-\frac{\varepsilon^2 \cdot \mu_H}{3+\varepsilon}\right) \quad \text{and} \quad \Pr\left(X < (1-\varepsilon) \cdot \mu_L\right) \leq \exp\left(-\frac{\varepsilon^2 \cdot \mu_L}{2+\varepsilon}\right).$$

A.2.2 Sketching Tools

We present the following sketching tool by Assadi and Shah [AS22] for the neighbourhood edge sampling problem.

Problem 1. Given a graph $G = (V, E)$ specified in a dynamic stream, and a set $S \subseteq V$ of vertices at the start of the stream, output an edge (u, v) such that $u \in S$ and v is sampled uniformly at random from $N(S)$.

We will use the following linear sketch for solving this problem.

Proposition A.3 ([AS22])**.** *There is a linear sketch, called* `NE-Sampler`(G, S)*, for Problem 1 with size*

$$s_{\texttt{NES}} = s_{\texttt{NES}}(n) = O(\log^3 n)$$

bits, that outputs FAIL *with probability at most* $1/100$ *and gives a wrong answer with probability at most* n^{-8}*.*

A.2.3 Proof of Lemma 7.5

Lemma 7.5. *There is a linear sketch for insertion-deletion graph streams that, given any graph $G = (V, E)$ specified via $\mathbf{vec}(E)$, uses $O(n^2/\alpha^2)$ bits of space and with high probability outputs a matching M_{easy} that satisfies at least one of the following conditions for any $\alpha \leq n^{1-\delta}$ and $\delta > 0$:*

- ***Match-case:*** *The matching M_{easy} has at least $\frac{n}{8\alpha}$ edges;*
- ***Sparsify-case:*** *The induced subgraph of G on vertices not matched by M_{easy}, denoted by G_R, has at most $20 \cdot \frac{n}{\alpha} \cdot \log^4 n$ edges.*

The algorithm in Lemma 7.5 samples $\approx n^2/\alpha^2 \cdot \log^3 n$ edges from the graph using a non-uniform distribution as follows: for each sample, first pick $\approx n/\alpha$ vertices S uniformly at random and then use `NE-Sampler` to sample an edge from S to a vertex of $N(S)$ chosen uniformly at random. Given the bound of $O(\log^3 n)$ bits on the size of sketches for `NE-Sampler`, the total space of the algorithm can be bounded by $O(n^2/\alpha^2)$ bits. In the recovery phase then, a greedy matching is computed over these sampled edges and is returned as M_{easy}. Note that in the analysis it will be helpful to think of the edges being recovered one by one and fed to the greedy matching algorithm. The key change we make here is in the number of edge samples taken and that we fix the parameter $\beta = n$ (see Section 7.3 for a discussion of β). Formally, we have Algorithm 6.

Algorithm 6 Altered Match-Or-Sparsify Lemma (Lemma 7.5)

Input: A dynamic graph stream σ for a n-vertex graph $G = (V, E)$, a small constant $\delta > 0$, and a positive integer $\alpha \leq n^{1-\delta}$
Output: A matching M_{easy} in G

Pre-processing:

1: Let $k := n/\alpha$ and $s := n^2/\alpha^2 \cdot \log^3 n$
2: **for** $i \in [2s]$ [1] **do**
3: Sample a pair-wise independent hash function $h_i : V \to [k]$
4: Set $V_i := \{v \in V \mid h_i(v) = 1\}$
5: Initialise $\mathbf{N}_i$ to be an instance of `NE-Sampler`(G, V_i)

Processing the stream:

5: Update each $\mathbf{N}_i$ using σ

Post-processing:

8: For all $i \in [2s]$, run the recovery algorithm of `NE-Sampler`(G, V_i) on $\mathbf{N}_i$ to get an output edge e_i (we write $e_i = \perp$ if the sampler outputs FAIL)
9: Let M_{easy} be the greedy matching over the sampled edges e_i.
10: **return** M_{easy}

We first bound the space of the algorithm.

Lemma A.4. *Algorithm 6 uses $O(n^2/\alpha^2)$ bits of space with high probability.*

Proof. In each step, Line 3 requires storing a pair-wise independent hash function which needs $O(\log n)$ bits of space by Proposition 7.2. Line 5 requires storing an `NE-Sampler` which needs $O(\log^3 n)$ bits by Proposition A.3. There are $2s = O(n^2/\alpha^2 \cdot \log^3 n)$ steps, so the total space $O(n^2/\alpha^2)$ bits. □

We now prove that the matching M_{easy} output by Algorithm 6 satisfies the guarantees of Lemma 7.5. For simplicity, we use similar notations and definitions as the work by Assadi and Shah [AS22].

Notation. For any $i \in [2s]$, let M_i be the set of edges included in M_{easy} in the first $i-1$ steps of the recovery, i.e., from $\{e_j\}_{j=1}^{i-1}$, and G_R^i to be the subgraph of G induced on *unmatched* vertices of M_i. We use $\deg_i(v)$ to denote the degree of each vertex in G_R^i to other vertices in G_R^i. We partition vertices of G_R^i based on their degrees in G_R^i into **low-**, **medium-**, and **high-degree** as follows:

$$\text{Low}_i := \left\{v : \deg_i(v) < (\log^3 n)\right\}, \ \text{Med}_i := \left\{v : (\log^3 n) \leq \deg_i(v) < (\frac{n}{8\alpha})\right\},$$
$$\text{High}_i := \left\{v : \deg_i(v) \geq \frac{n}{8\alpha}\right\}.$$

We define the following two events:

- $\mathcal{E}_{\text{M}}(i)$: the matching M_i has less than $(n/8\alpha)$ edges (i.e., matching-case not happened);
- $\mathcal{E}_{\text{S}}(i)$: the subgraph G_R^i has more than $(20\, n \cdot \log^4 n/\alpha)$ edges (i.e., sparsify-case not happened).

Finally, we say that a choice of V_i in step $i \in [2s]$ is **clean** if V_i does not contain any matched vertices of M_i.

The key change we make here is that, in our partitioning of G_R^i, the boundary between **low-** and **medium-** is reduced by an α-factor. Coupled with the increase in number of samples, this allows us to also change the definition of event $\mathcal{E}_{\text{S}}(i)$ to reflect the α-factor increase in sparseness guarantees which we require.

The events are defined in such a way that if at least one of these events do not happen for some $i \in [2s]$, then Algorithm 6 succeeds in outputting the desired matching of Lemma 7.5. The formal argument is as follows.

[1] The steps are partitioned into two **batches** of s steps each in the analysis, hence the use of $2s$ for the number of steps.

Claim A.5. *Suppose for some $i \in [2s]$, either of $\mathcal{E}_{\mathrm{M}}(i)$ or $\mathcal{E}_{\mathrm{S}}(i)$ does not happen; then, M_{easy} of Algorithm 6 satisfies the guarantees of Lemma 7.5.*

Proof. Suppose first that $\mathcal{E}_{\mathrm{M}}(i)$ does not happen. This means M_i has size at least $(n/8\alpha)$ and by the greedy choice of M_{easy}, we have $|M_{\mathrm{easy}}| \geq |M_i| \geq (n/8\alpha)$, satisfying the match-case condition.

Now suppose that $\mathcal{E}_{\mathrm{S}}(i)$ does not happen. Since the number of edges of G_R can only be smaller than that of G_R^i, we have that G_R also only has $(20\, n \cdot \log^4 n/\alpha)$ edges, satisfying the sparsify-case condition. □

The idea is to show that with high probability, for some $i \in [2s]$, one of the events $\mathcal{E}_{\mathrm{M}}(i)$ or $\mathcal{E}_{\mathrm{S}}(i)$ is *not* going to happen, then Lemma 7.5 holds by Lemma A.4 and Claim A.5. In order to do this, we need to show that, for any step i, there will be a probability of $\approx k/s$ in increasing the size of M_i by one as long as both $\mathcal{E}_{\mathrm{M}}(i)$ and $\mathcal{E}_{\mathrm{S}}(i)$ happen. Therefore, if these events both happen for at least s steps, it will ultimately lead to event $\mathcal{E}_{\mathrm{M}}(2s)$ not happening, i.e., a large matching M_{easy}.

The variance reduction ideas used to prove this rely on the set High_i being empty for each step i. Hence, to achieve this, the steps of the algorithm are partitioned into two **batches** each of size s. The analysis of the first batch shows that High_i is empty for every step in the second batch, i.e., for all $i \in (s, 2s]$. Then, the second batch is used for the main argument. We will prove the following:

- **First batch:** As long as $\mathcal{E}_{\mathrm{M}}(i)$ and $\mathcal{E}_{\mathrm{S}}(i)$ happen for all $i \in [s]$, with high probability, the set High_{s+1} (and thus High_j for all $j \in (s, 2s]$) will be empty for the second batch.

  Lemma A.6. *With high probability, either at least one of $\mathcal{E}_{\mathrm{M}}(i)$ and $\mathcal{E}_{\mathrm{S}}(i)$ does not happen for some step $i \in [s]$ or High_{s+1} will be empty.*

- **Second batch:** Whenever both $\mathcal{E}_{\mathrm{M}}(i)$ and $\mathcal{E}_{\mathrm{S}}(i)$ happen in a step $i \in (s, 2s]$, there will be a probability of $\approx k/s$ in increasing the size of M_i by one in this step. Given that this process is repeated for s steps, M_{easy} will eventually become of size $\approx k = n/\alpha$ (or one of the events happen along the way, and Claim A.5 is used instead).

  Lemma A.7. *Assuming High_{s+1} is empty, with high probability, at least one of the events $\mathcal{E}_{\mathrm{M}}(i)$ or $\mathcal{E}_{\mathrm{S}}(i)$ does not happen for some $i \in (s : 2s]$.*

We now make the following remark.

Remark A.8. The actions of Algorithm 6 are clearly *not* independent across different steps (in the recovery phase). However, in the upcoming probability analysis in each step $i \in [2s]$ the randomness of all prior steps conditioned on the events $\mathcal{E}_{\mathrm{M}}(i)$ and $\mathcal{E}_{\mathrm{S}}(i)$ are fixed, and *only*

the randomness of the choice of (V_i, e_i) are used in this step. This randomness is independent of prior steps. As such, in the following, **all probability calculations in a step i are conditioned on randomness of prior steps and events $\mathcal{E}_{\text{M}}(i)$ and $\mathcal{E}_{\text{S}}(i)$**, without writing it explicitly each time. These probability calculations may not necessarily remain correct when either of these events do not happen, but we will be done by Claim A.5 in those cases anyway.

The following simple helper claim will be useful in the subsequent proofs (this claim would have been trivial had h_i been a truly independent hash function instead of a pairwise-independent one).

Claim A.9. *Consider any step $i \in [s]$ and let v be any arbitrary vertex in G_R^i. Then,*

$$\Pr_{V_i}\Big(v \in V_i \text{ and } V_i \text{ is clean}\Big) \geq \frac{3}{4k}.$$

Proof. Recall that there are at most $n/4\alpha = k/4$ vertices matched by M_i. We have,

$$\begin{aligned}
\Pr_{V_i}\Big(v \in V_i \text{ and } V_i \text{ is clean}\Big) &= \Pr\left(h_i(v) = 1\right) \cdot \Pr\left(V_i \text{ is clean} \mid h_i(v) = 1\right) \\
&\qquad (v \in V_i \text{ iff } h_i(v) = 1) \\
&= \frac{1}{k} \cdot \Big(1 - \Pr\left(V_i \text{ is not clean} \mid h_i(v) = 1\right)\Big) \\
&\qquad (\text{as } h_i(v) = 1 \text{ w.p. } 1/k) \\
&\geq \frac{1}{k} \cdot \left(1 - \sum_{u \in V(M_i)} \Pr\left(h_i(u) = 1 \mid h_i(v) = 1\right)\right) \\
&\qquad (\text{by union bound and since } V_i \text{ is not clean iff } h_i(u) = 1 \text{ for some } u \in V(M_i)) \\
&= \frac{1}{k} \cdot \left(1 - \sum_{u \in V(M_i)} \frac{1}{k}\right) \\
&\qquad (h_i(\cdot) \text{ is a pairwise-independent hash function}) \\
&= \frac{1}{k} \cdot \left(1 - \frac{k}{4} \cdot \frac{1}{k}\right), \\
&\qquad (\text{as } h_i(u) = 1 \text{ w.p. } 1/k \text{ and there are at most } k/4 \text{ choices for matched vertices})
\end{aligned}$$

which is at least $3/4k$ as desired. □

In Appendix A.2.4, we follow exactly the steps taken by Assadi and Shah [AS22] and get the exact same intermediary results. The difference is crucially in the final arguments which rely on the number of samples s taken. Their analysis rely on the assumption that $n \geq \alpha^2 \cdot n^{\delta}$ which does not hold in our case. This, however, is circumvented by the additional α-factor in the number of samples taken.

In Appendix A.2.5, we also follow their analysis closely; however, the change in definition of $\mathcal{E}_{\text{S}}(i)$ for each step i slightly alters the intermediary results. In essence, the probability that

we increase the size of the matching in a particular step ($\approx k/s$) can be an α-factor smaller since we have an α-factor many more steps.

A.2.4 First Batch: Proof of Lemma A.6

Let v be any vertex in V and consider any step $i \in [s]$. If $\deg_i(v) < (n/8\alpha)$, then v cannot be part of High_i and subsequently High_{s+1} since G_R^{s+1} is a subgraph of G_R^i. In the following, we consider the case where $\deg_i(v) \geq (n/8\alpha)$ and prove that there is a non-trivial chance of "progress" (to be defined later) in each step. We first bound the probability of the following useful event for our analysis.

Claim A.10. *In step i, if $\deg_i(v) \geq n/8\alpha$, we have $\Pr_{V_i}(v \in N(V_i) \text{ and } V_i \text{ is clean}) \geq \frac{1}{16}$.*

Proof. Let $d(v) := n/8\alpha$ and $D(v)$ be a set of d arbitrary neighbors of v in G_R^i. We know that v will be included in $N(V_i)$ if any of vertices in $D(v)$ is sampled in V_i. We have,

$$\begin{aligned}
\Pr_{V_i}(v \in N(V_i) \text{ and } V_i \text{ is clean}) &\geq \Pr(D(v) \cap V_i \neq \emptyset \text{ and } V_i \text{ is clean}) && (D(v) \subseteq N(v)) \\
&\geq \sum_{u \in D(v)} \Pr(u \in V_i \text{ and } V_i \text{ is clean}) - \sum_{u \neq w \in D(v)} \Pr(u, w \in V_i) \\
&\text{(by inclusion-exclusion principle and bounding } \Pr(u, w \in V_i) \geq \Pr(u, w \in V_i \text{ and } V_i \text{ is clean})) \\
&> \frac{3d(v)}{4k} - \frac{d(v)^2}{k^2} \\
&\text{(by Claim A.9 and as } h_i(\cdot) \text{ is a pair-wise independent hash function with range } [k]) \\
&= \frac{(n/8\alpha)}{(n/\alpha)} \cdot \left(\frac{3}{4} - \frac{(n/8\alpha)}{(n/\alpha)}\right), && (\text{as } d(v) = n/8\alpha \text{ and } k = n/\alpha)
\end{aligned}$$

which is at least $1/16$ as desired. □ Claim A.10

Let us now condition on the choice of V_i and assume the event of Claim A.10 has happened. We say that this step i is a **matching-step** if $N(V_i) > (n/2\alpha)$; otherwise, we call this step a **vertex-step**. We argue that in a matching-step we have a constant probability of increasing the size of M_i by one and in a vertex-step we have a probability $\approx \alpha/n$ of matching the vertex v and thus no longer including it in G_R^{i+1} and High_{i+1}. We formalize this in the following.

Claim A.11. *Fix V_i and suppose step i is a matching-step and the event of Claim A.10 has happened. Then,*

$$\Pr_{e_i}(e_i \in M_{i+1} \mid V_i) \geq \frac{1}{3}.$$

Proof. As $N(V_i)$ contains more than $(n/2\alpha)$ vertices (as this is matching-step) while M_{easy} has at most $(n/4\alpha)$ vertices (as $\mathcal{E}_{\text{M}}(i)$ has happened), we know that at least half the vertices in

$N(V_i)$ are unmatched. Given that all of V_i is also unmatched, if $\texttt{NE-Sampler}(G, V_i)$ samples e_i to any of the unmatched vertices in $N(V_i)$, we can include e_i in M_{i+1} greedily. As the choice of $\texttt{NE-Sampler}(G, V_i)$ is uniform over $N(V_i)$, this event happens with probability at least $(1/2 - \delta_F) > 1/3$, as desired (since $\delta_F = 1/100$). □ Claim A.11

Claim A.12. *Fix V_i and suppose step i is a vertex-step and the event of Claim A.10 has happened. Then,*

$$\Pr_{e_i}(v \in V(M_{i+1}) \mid V_i) \geq \frac{\alpha}{n}.$$

Proof. We know $v \in N(V_i)$ and that size of $N(V_i)$ is at most $(n/2\alpha)$. At the same time, since V_i is clean, if v is sampled as an endpoint of e_i by $\texttt{NE-Sampler}(G, V_i)$, the edge e_i will join the matching greedily and thus v will be matched. As the choice of $\texttt{NE-Sampler}(G, V_i)$ is uniform over $N(V_i)$ and $\delta_F = 1/100$,

$$\Pr_{e_i}(v \in V(M_{i+1}) \mid V_i) \geq (1 - \delta_F) \cdot \frac{1}{|N(V_i)|} > \frac{\alpha}{n}. \quad \square \text{ Claim A.12}$$

□

We can now conclude the proof of Lemma A.6 as follows. We have that at least half the steps are matching-steps or half of them are vertex-steps. We consider each case as follows.

When half the steps are matching-steps. In this case, each matching-step i increases size of M_i by one with probability at least $(1/48)$ by Claims A.10 and A.11. Thus,

$$\mathbb{E}|M_{s+1}| \geq (\frac{s}{2}) \cdot \frac{1}{48} = \frac{n^2}{2\alpha^2 \cdot \log^3 n} \cdot \frac{1}{48} \gg n/\alpha,$$

given that $\alpha \leq n^{1-\delta}$. Moreover, the distribution of M_{s+1} statistically dominates sum of $(s/2)$ Bernoulli random variables with mean $(1/48)$. As such, by the Chernoff bound (Proposition A.2),

$$\Pr(M_{s+1} < (n/8\alpha)) < \exp(-n/\alpha) \ll 1/\operatorname{poly}(n),$$

as $\alpha \leq n^{1-\delta}$. This implies that $\mathcal{E}_{\mathrm{M}}(s+1)$ happens, proving Lemma A.6 in this case.

When half the steps are vertex-steps. In this case, each vertex-step i can independently match the vertex v with probability at least $(\alpha/16\,n)$ by Claims A.10 and A.12. Thus,

$$\Pr(v \in \text{High}_{s+1}) \leq (1 - \frac{\alpha}{16\,n})^{s/2} \leq \exp\left(-\frac{\alpha}{16\,n} \cdot \frac{n^2}{2 \cdot \alpha^2 \cdot \log^3 n}\right) < \exp\left(-\frac{n^{\delta/2}}{32}\right) \ll 1/\operatorname{poly}(n),$$

where we use $\alpha \leq n^{1-\delta}$. Thus, with high probability v will not be part of High_{s+1}. A union bound over all the vertices $v \in V$ then ensures that High_{s+1} will be empty with high probability, thus proving Lemma A.6 in this case too.

Remark: We note that the definition of matching-steps and vertex-steps are tailored to individual vertices in V; however, even if one vertex leads to having at least half of the steps as matching-steps, we can apply the argument of first part and conclude the proof. Thus, when applying the second part of the argument, we can assume that all vertices lead to half of the steps being vertex-steps, and so we can union bound over all of them.

A.2.5 Second Batch: Proof of Lemma A.7

We now prove Lemma A.7. In the following, we condition on the event that High_{s+1} (and High_i for every $i \in (s, 2s]$) is empty. Our goal is then to prove that at some step $i \in (s, 2s]$, one of the events $\mathcal{E}_{\text{M}}(i)$ or $\mathcal{E}_{\text{S}}(i)$ is not going to happen. The key to the proof of Lemma A.7 (and Lemma 7.5 itself) is the following.

Lemma A.13. *For any $i \in (s\,,\,2s]$,*

$$\Pr_{(V_i,e_i)}\Big(M_{i+1} > M_i\Big) \geq \frac{\alpha \cdot \log^3 n}{4 \cdot n}.$$

We first identify a simple structure in the graph G^i_R. The following claim is based on a standard low-degree orientation of the graph plus geometric grouping of degrees of vertices.

Claim A.14. *At least one of the following two conditions is true about G^i_R:*

(i) for some $d \in \left[\log^3 n, \frac{n}{8\alpha}\right)$, there are $\left(\frac{n \cdot \log^3 n}{2\alpha d}\right)$ vertices v in Med_i with $\deg_i(v) \geq d$;

(ii) for some $d \in [1, \log^3 n)$, there are $\left(\frac{19\, n \cdot \log^3 n}{2\alpha d}\right)$ vertices in Low_i with at least d neighbors in Low_i.

Proof. Given that High_i is empty, any edge in G^i_R is either incident on Med_i or is between two vertices in Low_i. Consequently, given that by $\mathcal{E}_{\text{S}}(i)$, we have at least $(20\,(n/\alpha) \cdot \log^4 n)$ edges in G^i_R, there are either at least $(n \cdot \log^4 n/\alpha)$ edges incident on Med_i or $(19 \cdot n \cdot \log^4(n)/\alpha)$ edges entirely inside Low_i. We prove that each case corresponds to one of the conditions in the claim.

When $\geq ((n/\alpha) \cdot \log^4 n)$ edges are incident on Med_i. We partition vertices in Med_i into sets $\{D_j\}$ where each D_j contains vertices v with $\deg_i(v) \in [2^j, 2^{j+1})$. As such,

$$\sum_j |D_j| \cdot 2^{j+1} \geq \#\text{ edges incident on } \text{Med}_i \geq (n/\alpha) \cdot \log^4 n.$$

As there are at most $\log n$ choices for j in the summation above, we should have some D_{j^*} with

$$|D_{j^*}| \geq \frac{(n/\alpha) \cdot \log^3 n}{2^{j^*+1}}.$$

Setting $d = 2^{j^*}$ and returning (a subset of) D_{j^*} satisfies the bound in part (i) of the claim: all vertices in $D_{j^*} \subseteq \text{Med}_i$ have $\deg_i(\cdot)$ in $[\log^3 n, \frac{n}{8\alpha})$ by definition of Med_i, and we can pick a subset of D_{j^*} with size prescribed by the claim as all vertices in D_{j^*} have degree d at least.

When $\geq (19\,(n/\alpha) \cdot \log^4 n)$ edges are entirely inside Low_i. The argument is almost identical to the above part by counting the degree of vertices in Low_i but only in Low_i (instead of all of $\deg_i(\cdot)$ as in the previous part). We partition vertices in Low_i into sets $\{D_j\}$ where each D_j contains all vertices with number of neighbors in Low_i in $[2^j, 2^{j+1})$. As such,

$$\sum_j |D_j| \cdot 2^{j+1} \geq \#\text{ edges entirely inside } \text{Low}_i \geq 19\,(n/\alpha) \cdot \log^4 n.$$

As there are at most $\log n$ choices for j in the summation above, we should have some D_{j^*} with

$$|D_{j^*}| \geq \frac{19\,(n/\alpha) \cdot \log^3 n}{2^{j^*+1}}.$$

Setting $d = 2^{j^*}$ and returning (a subset of) D_{j^*} satisfies the bound in part (ii) of the claim: all vertices in $D_{j^*} \subseteq \text{Low}_i$ have degree less than $(\log^3 n)$ by the definition of Low_i (even in G^i_R and so between Low_i also) and we can pick a subset of D_{j^*} with the required size as vertices in D_{j^*} have degree d at least. □ Claim A.14

In the following, we refer to a step $i \in (s, 2s]$ as a **V_i-step** whenever case (i) of Claim A.14 happens and a **$N(V_i)$-step** otherwise. We will show that:

- In a **V_i-step**, we have "enough" large degree vertices and even if we sample one of them in V_i it will make the intersection of $N(V_i)$ and G^i_R large;
- In a **$N(V_i)$-step**, we have "so many" low degree vertices in G^i_R that many of them will appear in $N(V_i)$ and thus there is a large intersection between $N(V_i)$ and G^i_R again.

In each case, we can finalize the proof by showing that having $N(V_i)$ intersect largely with G^i_R allows us to recover an edge e_i via $\texttt{NE-Sampler}(G, V_i)$ that can increase size of M_i with sufficiently large probability.

Case (i) of Claim A.14: V_i-steps

Let

$$d \in [\log^3 n, \frac{n}{8\alpha}) \quad \text{and} \quad D \subseteq \text{Med}_i \quad \text{with} \quad |D| = \frac{n \cdot \log^3 n}{2\alpha d} \tag{A.3}$$

be, respectively, the degree-parameter and corresponding set guaranteed by Case (i) of Claim A.14. The following claim lower bounds the probability that V_i is both clean and samples a vertex from D.

Claim A.15. $\Pr_{V_i}(V_i \cap D \neq \emptyset \text{ and } V_i \text{ is clean}) \geq \frac{\log^3 n}{8d}$.

Proof. We have,

$$\Pr(V_i \cap D \neq \emptyset \text{ and } V_i \text{ is clean}) \geq \sum_{v \in D} \Pr(v \in V_i \text{ and } V_i \text{ is clean}) - \sum_{u \neq w \in D} \Pr(u, w \in V_i)$$

(by inclusion-exclusion principle and dropping the 'intersection' from the second event)

$$\geq |D| \cdot \frac{3}{4k} - |D|^2 \cdot \frac{1}{k^2}$$

(by Claim A.9 and as $h_i(\cdot)$ is a pair-wise independent hash function with range $[k]$)

$$= \frac{n \cdot \log^3 n}{2\alpha d} \cdot \frac{\alpha}{n} \cdot \left(\frac{3}{4} - \frac{n \log^3 n \cdot \alpha}{2\alpha d \cdot n}\right)$$

(by the choice of $k = n/\alpha$ and size of D in Equation (A.3))

$$\geq \frac{\log^3 n}{8d},$$

as $d \geq \log^3 n$ by Equation (A.3). □ Claim A.15

Let us now condition on the choice of V_i and assume the event of Claim A.15 happens. Given that any vertex in D already has d neighbors in G_R^i, we have that $N(V_i) \cap G_R^i$ has size at least d in this case. On the other hand, $N(V_i)$ can have at most $(n/4\alpha)$ neighbors outside G_R^i by the bound on the total number of matched vertices by $\mathcal{E}_{\mathrm{M}}(i)$. As the choice of e_i from $\texttt{NE-Sampler}(G, V_i)$ is uniform over $N(V_i)$, we have,

$$\Pr_{e_i}\left(e_i \text{ is from } V_i \text{ to } N(V_i) \cap G_R^i \mid V_i\right) \geq (1 - \delta_F) \cdot \frac{|N(V_i) \cap G_R^i|}{|N(V_i)|}$$

$$\geq (1 - \delta_F) \cdot \frac{d}{(n/4\alpha) + d}$$

$$\geq (1 - \delta_F) \cdot \frac{d \cdot 8\alpha}{3n} \quad (\text{as } d \leq (n/8\alpha) \text{ in Equation (A.3)})$$

$$\geq \frac{2d \cdot \alpha}{n},$$

as $\delta_F < 1/4$. Given that all of V_i is also unmatched (as V_i is clean by conditioning on the event of Claim A.15), we can include e_i in M_{i+1} greedily whenever e_i is between V_i and $N(V_i) \cap G_R^i$.

Consequently, combining the two events above, we have,

$$\Pr_{(V_i, e_i)}\left(M_{i+1} > M_i\right) \geq \Pr_{V_i}(V_i \cap D \neq \emptyset \text{ and } V_i \text{ is clean}) \cdot \Pr_{e_i}\left(e_i \text{ is from } V_i \text{ to } N(V_i) \cap G_R^i \mid V_i\right)$$

$$\geq \frac{\log^3 n}{8d} \cdot \frac{2d \cdot \alpha}{n} = \frac{\alpha \log^3 n}{4\, n}.$$

This concludes the proof of Lemma A.13 in this case.

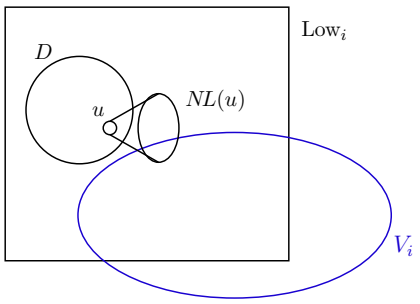
Case (ii) of Claim A.14: $N(V_i)$-steps

Let

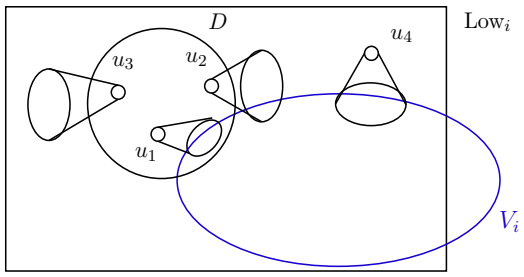
$$d \in [1, \log^3 n) \quad \text{and} \quad D \subseteq \text{Low}_i \quad \text{with} \quad |D| = \frac{19\,(n/\alpha) \cdot \log^3 n}{2d} \tag{A.4}$$

be, respectively, the degree-parameter and corresponding set guaranteed by Case (ii) of Claim A.14. For the rest of this analysis, we focus only on the subgraph of G_R^i induced on vertices of Low_i and for each $v \in D$, we pick exactly d (arbitrary) neighbors from Low_i and denote them by $NL(v)$. Our goal is to show that $N(V_i)$ and G_R^i intersect largely. We will do so by counting the elements in D that have neighbors in V_i. This works because $D \subseteq G_R^i$ and having neighbors in V_i means that the vertex itself is in $N(V_i)$.

For any vertex $v \in D$, define an indicator random variable $X_v \in \{0, 1\}$ which is 1 iff $NL(v) \cap V_i \neq \emptyset$ (see Figure A.1a). Notice that $X = \sum_{v \in D} X_v$ is a random variable that denotes the number of vertices v in D that have a neighbor in $NL(v)$ that belongs to V_i. Note that we do not consider all neighbors of v in Low_i, only the ones in $NL(v)$; this is okay since we just need a lower bound on $\left|N(V_i) \cap G_R^i\right|$. It is easy to see that $X \leq \left|N(V_i) \cap G_R^i\right|$ since v contributes to $\left|N(V_i) \cap G_R^i\right|$ if $X_v = 1$ (see Figure A.1b). We first bound the probability of the event $NL(v) \cap V_i \neq \emptyset$.



(a) This figure shows the set of vertices Low_i and its subset D. V_i (in blue) is the set of sampled vertices in step i. For a vertex u in D we define a set of neighbors $NL(u)$ which if intersects with V_i then we have random variable $X_u = 1$.



(b) This figure shows $X \leq \left|N(V_i) \cap G_R^i\right|$. The vertices u_j are in Low_i. Notice that $X_{u_1} = 1, X_{u_2} = 1, X_{u_3} = 0$ and X_{u_4} is not defined so we have $X = 2$. But $\left|N(V_i) \cap G_R^i\right| = 3$ since u_1, u_2 and u_4 contribute to it.

Figure A.1: Illustration of random variables X_v.

Claim A.16. *For any $v \in D$,*

$$(1 - o(1)) \cdot \frac{d \cdot \alpha}{n} \leq \Pr\left(X_v = 1\right) \leq \frac{d \cdot \alpha}{n}.$$

Proof. $X_v = 1$ iff one of the neighbors of v in $NL(v)$ belongs to V_i. For the upper bound, by

union bound,

$$\Pr\left(X_v = 1\right) \leq \sum_{u \in NL(v)} \Pr\left(u \in V_i\right) = |NL(v)| \cdot \frac{1}{k} = \frac{d \cdot \alpha}{n}. \qquad \text{(as } |NL(v)| = d \text{ and } k = n/\alpha\text{)}$$

For the lower bound, by inclusion-exclusion principle,

$$\begin{aligned}
\Pr\left(X_v = 1\right) &\geq \sum_{u \in NL(v)} \Pr\left(u \in V_i\right) - \sum_{u \neq w \in NL(v)} \Pr\left(u, w \in V_i\right) \\
&\geq |NL(v)| \cdot \frac{1}{k} - |NL(v)|^2 \cdot \frac{1}{k^2} \\
&= \frac{d \cdot \alpha}{n} \cdot \left(1 - \frac{d \cdot \alpha}{n}\right) && \text{(as } |NL(v)| = d \text{ and } k = n/\alpha\text{)} \\
&\geq (1 - o(1)) \cdot \frac{d \cdot \alpha}{n},
\end{aligned}$$

as $d < \log^3 n$ by Equation (A.4) and $\alpha \leq n^{1-\delta}$. □ Claim A.16

By Claim A.16 and the size of D in Equation (A.4), we have,

$$(1 - o(1)) \cdot \frac{19}{2} \cdot \log^3 n \leq \mathbb{E}\left[X\right] \leq \frac{19}{2} \cdot \log^3 n. \tag{A.5}$$

Our goal now is to prove that X is concentrated. This requires a non-trivial proof as the variables $\{X_v\}_{v \in D}$ are correlated through their shared neighbors in V_i. But the fact that the subgraph induced on Low_i is low-degree allows us to bound the variance of X using a combinatorial argument in the following claim.

Claim A.17. $\text{Var}\left[X\right] \leq (1/8) \cdot \mathbb{E}\left[X\right]^2$.

Proof. For any two vertices $u \neq v \in D$, define $\text{Com}(u, v) := NL(u) \cap NL(v)$ as the set of common neighbors of u and v in subgraph of Low_i defined by $NL(\cdot)$ and let $\text{com}(u, v) = |\text{Com}(u, v)|$. We have,

$$\text{Var}\left[X\right] = \sum_{v \in D} \text{Var}\left[X_v\right] + \sum_{u \neq v \in D} \text{Cov}\left[X_u, X_v\right] \leq \mathbb{E}\left[X\right] + \sum_{u \neq v \in D} \text{Cov}\left[X_u, X_v\right], \tag{A.6}$$

as X_v is an indicator random variable and thus $\text{Var}\left[X_v\right] \leq \mathbb{E}\left[X_v\right]$. We thus need to bound the covariance-terms only. Recall that

$$\begin{aligned}
\text{Cov}\left[X_u, X_v\right] &= \mathbb{E}\left[X_u \cdot X_v\right] - \mathbb{E}\left[X_u\right]\mathbb{E}\left[X_v\right] \\
&= \Pr\left(NL(u) \cap V_i \neq \emptyset \wedge NL(v) \cap V_i \neq \emptyset\right) - \Pr\left(X_u = 1\right) \cdot \Pr\left(X_v = 1\right).
\end{aligned}$$

We can bound the second part using Claim A.16 for each probability-term. For the first part, notice that for $NL(u) \cap V_i \neq \emptyset$ and $NL(v) \cap V_i \neq \emptyset$ one of the following two things should

happen: at least one of the shared neighbors of u, v in $\text{Com}(u,v)$ is chosen in V_i or each of them separately have a neighbor in $NL(u) - \text{Com}(u,v)$ and $NL(v) - \text{Com}(u,v)$ those join V_i (as $h_i(\cdot)$ is a pair-wise independent hash function, the probability of these two distinct vertices joining V_i is independent). Thus,

$$\begin{aligned}\Pr\left(NL(u) \cap V_i \neq \emptyset \wedge NL(v) \cap V_i \neq \emptyset\right) &\leq \sum_{w \in \text{Com}(u,v)} \Pr\left(w \in V_i\right) + \sum_{\substack{z_u \in NL(u) - \text{Com}(u,v) \\ z_v \in NL(v) - \text{Com}(u,v)}} \Pr\left(z_u \in V_i\right) \cdot \Pr\left(z_v \in V_i\right) \\ &\leq \text{com}(u,v) \cdot \frac{1}{k} + d^2 \cdot \frac{1}{k^2} \\ &\text{(as } h_i(\cdot) \text{ is uniform over } [k] \text{ and } u, v \in D \text{ and each vertex in } D \text{ has exactly } d \text{ neighbors in } NL(\cdot)) \\ &= \text{com}(u,v) \cdot \frac{\alpha}{n} + \frac{d^2 \cdot \alpha^2}{n^2}. \qquad \text{(as } k = n/\alpha)\end{aligned}$$

Plugging in this for the first term of covariance and the bounds in Claim A.16 for the second terms, we have,

$$\text{Cov}\left[X_u, X_v\right] \leq \text{com}(u,v) \cdot \frac{\alpha}{n} + \frac{d^2 \cdot \alpha^2}{n^2} - (1 - o(1)) \cdot \frac{d^2 \cdot \alpha^2}{n^2} = \text{com}(u,v) \cdot \frac{\alpha}{n} + o(1) \cdot \frac{d^2 \cdot \alpha^2}{n^2}.$$

By plugging in further in the RHS of Equation (A.6), we get that,

$$\begin{aligned}\text{Var}\left[X\right] &\leq \mathbb{E}\left[X\right] + \left|D^2\right| \cdot o(1) \cdot \frac{d^2 \cdot \alpha^2}{n^2} + \sum_{u \neq v \in D} \text{com}(u,v) \cdot \frac{\alpha}{n} \\ &\leq \mathbb{E}\left[X\right] + \left(\frac{19 \cdot (n/\alpha) \cdot \log^3 n}{2d}\right)^2 \cdot o(1) \cdot \frac{d^2 \cdot \alpha^2}{n^2} + \sum_{u \neq v \in D} \text{com}(u,v) \cdot \frac{\alpha}{n} \\ &\qquad \text{(by the bound on size of } D \text{ in Equation (A.4))} \\ &\leq o(1) \cdot \mathbb{E}\left[X\right]^2 + \frac{\alpha}{n} \cdot \sum_{u \neq v \in D} \text{com}(u,v). \\ &\qquad \text{(by the lower bound on } \mathbb{E}\left[X\right] \geq (1 - o(1)) \cdot (19/2) \cdot \log^3 n \text{ in Equation (A.5))}\end{aligned}$$

The remaining part is then to compute the summation in the RHS which we do below using a double-counting argument. Note that $\sum_{u \neq v \in D} \text{com}(u,v)$ counts the number of common neighbors inside $NL(\cdot)$-subgraph of Low_i for each pair of vertices in D. This can be alternatively counted by going over vertices in Low_i that are neighbor to D and count the number of pairs of neighbors (in $NL(\cdot)$) they have in D.

$$\begin{aligned}\sum_{u \neq v \in D} \text{com}(u,v) &= \sum_{z \in NL(D)} \binom{|NL(z)|}{2} \leq \sum_{z \in NL(D)} |NL(z)|^2 \\ &\leq (\log^3 n) \cdot \sum_{z \in NL(D)} |NL(z)| \\ &\text{(each vertex in } \text{Low}_i \text{ has degree at most } (\log^3 n) \text{ to } \text{Low}_i \text{ and } z \in \text{Low}_i \text{ as it is in } NL(D)) \\ &\leq (\log^3 n) \cdot |D| \cdot d \\ &\text{(as the sum-term counts the number of edges between } D \text{ and } NL(D) \text{ which is at most } |D| \cdot d)\end{aligned}$$

$$\leq (\log^3 n) \cdot (\frac{19}{2} \cdot (n/\alpha) \cdot \log^3 n). \qquad \text{(by Equation (A.4) on the size of } D)$$

Plugging in this bound in the upper bound on $\text{Var}[X]$ in the earlier equation, we have,

$$\begin{aligned}\text{Var}[X] &\leq o(1) \cdot \mathbb{E}[X]^2 + \frac{\alpha}{n} \cdot \left(\frac{19}{2} \cdot (n/\alpha) \cdot \log^6 n\right) \\ &= o(1) \cdot \mathbb{E}[X]^2 + \frac{19}{2} \cdot \log^6 n \\ &< \frac{1}{8} \cdot \mathbb{E}[X]^2,\end{aligned}$$

as $\mathbb{E}[X] \geq (1 - o(1)) \cdot \frac{19}{2} \cdot (\log^3 n)$ by Equation (A.5). □ Claim A.17

Recall that

$$\left|N(V_i) \cap G_R^i\right| \geq X.$$

Given the bound on expectation and variance of X in Equation (A.5) and Claim A.17, respectively, we can now apply Chebyshev's inequality and get that,

$$\Pr\left(\left|N(V_i) \cap G_R^i\right| < 2 \cdot \log^3 n\right) \leq \Pr\left(|X - \mathbb{E}[X]| \geq \frac{15}{19} \cdot \mathbb{E}[X]\right) \leq \frac{19^2 \cdot \text{Var}[X]}{15^2 \cdot \mathbb{E}[X]^2} \leq \frac{19^2}{15^2 \cdot 8} < \frac{1}{4}.$$

Additionally, we also have that the probability that V_i is not clean is at most,

$$\Pr(V_i \text{ is not clean}) \leq \sum_{u \in V(M_i)} \Pr(u \in V_i) < \frac{n}{4\alpha} \cdot \frac{\alpha}{n} = \frac{1}{4}.$$

By a union bound on the two equations above, we have,

$$\Pr\left(\left|N(V_i) \cap G_R^i\right| \geq 2\log^3 n \text{ and } V_i \text{ is clean}\right) \geq \frac{1}{2}. \tag{A.7}$$

The rest of the proof is similar to that of **V_i-steps**. We condition the choice of V_i and assume the event of Equation (A.7) has happened. Thus, we have that both V_i is clean and $N(V_i)$ has at least $2\log^3 n$ vertices in G_R^i. Moreover, $N(V_i)$ can have at most $(n/4\alpha)$ neighbors outside G_R^i by the bound on the total number of matched vertices by $\mathcal{E}_{\text{M}}(i)$. As the choice of e_i from `NE-Sampler`(G, V_i) is uniform over $N(V_i)$, we have,

$$\begin{aligned}\Pr_{e_i}\left(e_i \text{ is from } V_i \text{ to } N(V_i) \cap G_R^i \mid V_i\right) &\geq (1 - \delta_F) \cdot \frac{\left|N(V_i) \cap G_R^i\right|}{|N(V_i)|} \\ &\geq (1 - \delta_F) \cdot \frac{2\log^3 n}{(n/4\alpha) + 2\log^3 n} \\ &> \frac{3 \cdot \alpha \cdot \log^3 n}{n},\end{aligned}$$

as $\alpha \leq n^{1-\delta}$ and $\delta_F < 1/4$. Given that all of V_i is also unmatched (as V_i is clean), we can include e_i in M_{i+1} greedily whenever the event of the LHS above happens.

Consequently, combining the two events above, we have,

$$\begin{aligned}\Pr_{(V_i,e_i)}\left(M_{i+1} > M_i\right) \geq &\Pr_{V_i}\left(\left|N(V_i) \cap G_R^i\right| \geq 2\log^3 n \text{ and } V_i \text{ is clean}\right) \cdot \\ &\Pr_{e_i}\left(e_i \text{ is from } V_i \text{ to } N(V_i) \cap G_R^i \mid V_i\right) \\ \geq &\frac{1}{2} \cdot \frac{3 \cdot \alpha \cdot \log^3 n}{n} > \frac{\alpha \log^3 n}{n}.\end{aligned}$$

This concludes the proof of Lemma A.13 in this case also.

Concluding the Proof of Lemma A.7

By Lemma A.13, assuming the events $\mathcal{E}_{\mathrm{M}}(i), \mathcal{E}_{\mathrm{S}}(i)$ hold for every $i \in (s, 2s]$, size of M_{2s} statistically dominates sum of s independent Bernoulli random variables $\{Z_i\}_{i=1}^{s}$ with mean $(\alpha \cdot \log^3 n/4n)$ (RHS of Lemma A.13). Let $Z = \sum_{i=1}^{s} Z_i$. Thus, by the choice of s in Algorithm 6,

$$\mathbb{E}[Z] = \frac{n^2}{\alpha^2 \cdot \log^3 n} \cdot \frac{\alpha \cdot \log^3 n}{4 \cdot n} = \frac{n}{4 \cdot \alpha},$$

and by the Chernoff bound (Proposition A.2),

$$\Pr\left(Z < \frac{n}{8\alpha}\right) < \Pr\left(Z < \frac{1}{2} \cdot \mathbb{E}[Z]\right) \leq \exp\left(-\frac{n}{12\alpha}\right) \ll 1/\operatorname{poly}(n),$$

where the final bound is by $\alpha \leq n^{1-\delta}$. This means that as long as $\mathcal{E}_{\mathrm{M}}(i), \mathcal{E}_{\mathrm{S}}(i)$ happen for all $i \in (s : 2s]$, with high probability we are going to end up with a matching M_{easy} of size at least $(n/8\alpha)$, which means $\mathcal{E}_{\mathrm{M}}(2s)$ does not happen as desired.

This concludes the proof of Lemma 7.5.