ALGORITHMS FOR BIPARTITE MATCHING PROBLEMS WITH CONNECTIONS TO SPARSIFICATION AND STREAMING

A DISSERTATION
SUBMITTED TO THE INSTITUTE FOR COMPUTATIONAL AND MATHEMATICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Mikhail Kapralov
August 2012

Abstract

The problem of finding maximum matchings in bipartite graphs is a classical problem in combinatorial optimization with a long algorithmic history. Graph sparsification is a more recent paradigm of replacing a graph with a smaller subgraph that preserves some useful properties of the original graph, perhaps approximately. Traditionally, sparsification has been used for obtaining faster algorithms for cut-based optimization problems.

The contributions of this thesis are centered around new algorithms for bipartite matching problems, in which, surprisingly, graph sparsification plays a major role, and efficient algorithms for constructing sparsifiers in modern data models.

In the first part of the thesis we develop sublinear time algorithms for finding perfect matchings in regular bipartite graphs. These graphs have been studied extensively in the context of expander constructions, and have several applications in combinatorial optimization. The problem of finding perfect matchings in regular bipartite graphs has seen almost 100 years of algorithmic history, with the first algorithm dating back to König in 1916 and an algorithm with runtime linear in the number of edges in the graph discovered in 2000. In this thesis we show that, even though traditionally the use of sparsification has been restricted to cut-based problems, in fact sparsification yields extremely efficient *sublinear time* algorithms for finding perfect matchings in regular bipartite graphs when the graph is given in adjacency array representation. Thus, our algorithms recover a perfect matching (with high probability) without looking the whole input. We present two approaches, one based on independent sampling and another on random walks, obtaining an algorithm that recovers a perfect matching in $O(n \log n)$ time, within $O(\log n)$ of output complexity, essentially closing the problem.

In the second part of the thesis we study the streaming complexity of maximum bipartite matching. This problem is relevant to modern data models, where the algorithm is constrained in space and is only allowed few passes over the input. We are interested in determining the best tradeoff between the space usage and the quality of the solution obtained. We first study the problem in the single pass setting. A central object of our study is a new notion of sparsification relevant to matching problems: we define the notion of an ϵ-matching cover of a bipartite graph as a subgraph that approximately preserves sizes of matchings between every two subsets of vertices, which can be viewed as a sparsifier for matching problems. We give an efficient construction of a sparse subgraph that we call a matching skeleton, which we show is a linear-size matching cover for a certain range of parameters (in fact, for $\epsilon > 1/2$). We then show that our sparsifier can be applied repeatedly while

maintaining a non-trivial approximation ratio in the streaming model with vertex arrivals, obtaining the first $1 - 1/e$ deterministic one-pass streaming algorithm that uses linear space for this setting. Further, we show that this is in fact best possible: no algorithm can obtain a better than $1 - 1/e$ approximation in a single pass unless it uses significantly more than quasilinear space. This is a rather striking conclusion since a $1-1/e$ approximation can be obtained even in the more restrictive online model for this setting. Thus, we show that streaming algorithms can get no advantage over online algorithms for this problem unless they use substantially more than quasilinear space.

Our impossibility results for approximating matchings in a single pass using small space exploit a surprising connection between the sparsifiers that we define and a family of graphs known as Ruzsa-Szemerédi graphs. In particular, we show that bounding the best possible size of ϵ-covers for general ϵ is essentially equivalent to determining the optimal size of an ϵ-Ruzsa-Szemerédi graph. These graphs have received significant attention due to applications in PCP constructions, property testing and additive combinatorics, but determining their optimal size still remains a challenging open problem.

Besides giving matching upper and lower bounds for single pass algorithms in the vertex arrival setting, we also consider the problem of approximating matchings in multiple passes. Here we give an algorithm that achieves a factor of $1-e^{-k}k^k/k! = 1-\frac{1}{\sqrt{2\pi k}}+o(1/k)$ in k passes, improving upon the previously best known approximation.

In the third part of the thesis we consider the concept of *spectral sparsification* introduced by Spielman and Teng. Here, we uncover a connection between spectral sparsification and spanners, i.e. subgraphs that approximately preserve shortest path distances. This connection allows us to obtain a quasilinear time algorithm for constructing spectral sparsifiers using approximate distance oracles and entirely bypassing linear system solvers, which was previously the only known way of constructing spectral sparsifiers in quasilinear time.

Finally, in the last part of the thesis we design an efficient implementation of cut-preserving sparsification in a streaming setting with edge deletions using only one pass over the data.

Acknowledgements

First and foremost, I would like to thank my advisor, Ashish Goel. I was very fortunate to start working with Ashish during my first year in iCME. Our conversations with Ashish during the past five years have ranged from very technical meetings to discussing interesting research directions and then to broader career advice. Ashish's advice on any of these questions has invariably been precise and illuminating. His enthusiasm and encouragement has been a great source of inspiration for me during this time.

Sanjeev Khanna, a co-author on most of the results in this thesis, has been almost a second advisor to me. Most of the results on matchings in this thesis have been obtained in collaboration with Ashish and Sanjeev. The hours we spent thinking about augmenting paths, sampling and other things related to matchings are among my best moments at Stanford.

I was fortunate to experience the research environment at MSR Silicon Valley during two internships with Rina Panigrahy, which helped diversify and shape my research interests. I am grateful to Rina for being a great mentor and collaborator. I would like to thank Frank McSherry and Kunal Talwar for many inspiring discussions during this time. Also, I am grateful to Bahman Bahmani, Debo Dutta, Ian Post, Rajendra Shinde and Jan Vondrák for fruitful collaborations.

My master's thesis advisor, Alexander Katsevich, has played a pivotal role in my academic journey. Sasha's infectious enthusiasm about mathematical research, advice and encouragement has given me momentum that I am very grateful for.

Time spent outside of work would not have been as much fun without my best friends, John Clark and Sohan Dharmaraja, with whom we shared many a good time. I am also grateful to Pranav Dandekar, Pierre-David Letourneau, Arvind Krishna, Ilya Brailovsky, Shane Lewin, Tiyu Wang, Alex Papanicolau, Andrew Tausz and Guillaume Troianowski for providing great company on numerous non-academic occasions. I am especially thankful to Pierre-David Letourneau for his refreshingly pointed opinions.

Finally, I owe a great debt of gratitude to my family for their unconditional love and support.

Contents

List of Figures

Chapter 1

Introduction

The contributions of this thesis are centered around bipartite matching problems, for which we give algorithms that crucially use graph sparsification. We also consider the problem of implementing sparsification primitives in modern data models such as streaming and Map/Reduce. In the following sections we give an overview of the three main topics that this thesis spans, and outline our results.

1.1 Matchings in regular bipartite graphs

Let $G = (U, V, E)$ denote a bipartite graph with n vertices and m edges. A subset $M \subseteq E$ of edges is a *matching* if no two edges in M share an endpoint. The problem of finding maximum matchings in bipartite graphs is a classical problem in combinatorial optimization. The best known algorithm for finding matchings in bipartite graphs was obtained by Hopcroft and Karp in 1974 [45] and achieves $O(m\sqrt{n})$ runtime. An algebraic approach was used in [74] to obtain an algorithm with runtime $O(n^\omega)$, where ω is the matrix multiplication constant, which is better than $O(m\sqrt{n})$ for sufficiently dense graphs.

In the first part of the thesis we consider the problem of finding perfect matchings in *regular* bipartite graphs. A bipartite graph $G = (U, V, E)$ is regular if the degree of each vertex is exactly d, so that $m = nd$, where $n = |U|$. As a consequence of regularity one has $|U| = |V|$. These graphs have been studied extensively in the context of expander constructions, and have several applications in combinatorial optimization. The problem of finding perfect matchings in regular bipartite graphs has seen almost 100 years of algorithmic history, with the first algorithm dating back to König in 1916 [61] and an algorithm with $O(m)$ runtime discovered in 2000 by Cole, Ost and Schirra[23]. The main interest of Cole, Ost, and Schirra was in edge coloring of general bipartite graphs of maximum degree d, where finding perfect matchings in regular bipartite graphs is an important subroutine. Finding perfect matchings in regular bipartite graphs is also closely related to the problem of finding a Birkhoff von Neumann decomposition of a doubly stochastic matrix [17, 91]. Chapter 2 contains a more detailed discussion of prior work on these problems.

While the linear time algorithm of Cole, Ost and Schirra may seem to be a natural stopping point for this problem, in Chapter 2 we give *sublinear time* algorithms for finding a perfect matching in a regular bipartite graph using sparsification and random walks. We show that, even though traditionally the use of sparsification has been restricted to cut-based problems, in fact sparsification yields extremely efficient *sublinear time* algorithms for finding perfect matchings in regular bipartite graphs when the graph is given in adjacency array representation. Thus, our algorithms recover a perfect matching (with high probability) without looking the whole input. We present two approaches, one based on independent sampling and another on random walks, obtaining an algorithm that recovers a perfect matching in $O(n \log n)$ time, within $O(\log n)$ of output complexity, essentially closing the problem. Our techniques also yield efficient algorithms for edge coloring bipartite multigraphs and finding a Birkhoff-von Neumann decomposition of doubly stochastic matrices.

1.2 Graph sparsification

Large scale graphs are now a widely used tool for representing real world data. Many modern applications, such as search engines or social networks, require supporting various queries on large-scale graphs efficiently. An important primitive is maintaining a succint representation that preserves certain properties of the graph.

Various properties of graphs have been considered from the point of view of obtaining succint approximate representations recently. Perhaps the first such example is given by the celebrated cut-preserving sparsifiers of Benczúr and Karger[16]. *Spectral sparsification* is a generalization of cut-preserving sparsification introduced by Spielman and Teng[82] that has been used in constructing efficient algorithms for solving symmetric diagonally dominant linear systems in near-linear time [86, 63, 65]. A substantial body of work in the theoretical computer science literature is devoted to obtaining succint representations of graphs that support approximate distance queries such as spanners and approximate distance oracles [89, 90, 70, 8, 9, 10, 11]. A more detailed discussion of cut sparsifiers is given in Chapters 2 and 6. Spectral sparsifiers and spanners are discussed further in Chapter 5.

In this thesis we both borrow from and contribute to various notions of sparsification as follows:

1. In Chapters 2 and 7 we exhibit a novel connection between cut-preserving sparsification and the structure of matchings in regular bipartite graphs, obtaining sublinear time algorithms for finding perfect matchings in such graphs.

2. In Chapters 3 and 4 we introduce and study a new notion of sparsification relavant to matching problems in general bipartite graphs. This yields upper and lower bounds for the problem of finding matchings in a single pass in the streaming model.

3. In Chapter 5 we exhibit a novel connection between spanners and spectral sparsification, obtaining efficient algorithms for spectral sparsification.

4. In Chapter 6 we show how to construct spanners in a single pass in the streaming model with edge deletions using small space.

1.3 The streaming model

The need to process modern massive data sets necessitates rethinking classical solutions to many combinatorial optimization problems from the point of view of space usage and type of access to the data that algorithms assume. Applications in domains such as processing web-scale graphs, network monitoring or data mining among many others prohibit solutions that load the whole input into memory and assume random access to it. The streaming model of computation has emerged as a more realistic model for processing modern data sets. In this model the input is given to the algorithm as a stream, possibly with multiple passes allowed. The goal is to design algorithms that require small space and ideally one or a small constant number of passes over the data stream to compute

a (often approximate) solution. For many problems with applications in network monitoring, it has been shown that space polylogarithmic in the size of the input is often sufficient to compute very good approximate solutions. On the other hand, even basic graph algorithms have been shown to require $\Omega(n)$ space in the streaming model[27], where n is the number of vertices. A common relaxation is to allow $O(n \cdot \text{polylog}(n))$ space, a setting often referred to as the *semi-streaming* model. An important extension of the streaming model is the *dynamic* streaming model, where edge deletions are allowed in addition to edge addition.

The problem of implementing fundamental graph algorithms in the streaming model has received significant attention recently. Efficient algorithms for constructing sparsifiers[2, 57], spanners[8] and approximating matchings [28, 68, 26, 3, 4, 62] are known. Beautiful recent results of [5] show how to implement several fundamental graph algorithms in the dynamic streaming model as well. Further discussion of these models is given in Chapter 3 for matchings and Chapter 6 for sparsification in the dynamic streaming model.

In this thesis, we contribute to the study of graph algorithms in the streaming model in the following ways. First, in Chapter 3 we define a new notion of graph sparsification that is relevant to approximating matchings in the streaming model and use it to give the first strong lower bounds for approximating matchings in the streaming model. Further, we prove an optimal impossibility result on the approximation ratio that single pass streaming algorithms can achieve if they are constrained to use quasilinear space (i.e. semi-streaming algorithms). We also give a simple algorithm that improves upon the best known approximation for matchings using $k \geq 1$ passes. Second, in Chapter 6 we give a single pass algorithm for constructing cut-preserving sparsifiers in the streaming model with edge deletions.

Parts of this thesis have been published as papers or manuscripts [38, 36, 35, 49, 39, 48].

Chapter 2

Matchings in regular bipartite graphs

A bipartite graph $G = (U, V, E)$ with vertex set $U \cup V$ and edge set $E \subseteq U \times V$ is said to be regular if every vertex has the same degree d. We use $m = nd$ to denote the number of edges in G and n to represent the number of vertices in U (as a consequence of regularity, U and V have the same size). Regular bipartite graphs have been the subject of much study. Random regular bipartite graphs represent some of the simplest examples of expander graphs [72]. These graphs are also used to model scheduling, routing in switch fabrics, and task-assignment problems (sometimes via edge coloring, as described below) [1, 23].

A regular bipartite graph of degree d can be decomposed into exactly d perfect matchings, a fact that is an easy consequence of Hall's theorem [18]. Finding a matching in a regular bipartite graph is a well-studied problem, starting with the algorithm of König in 1916, which is now known to run in time $O(mn)$ [61]. The well-known bipartite matching algorithm of Hopcroft and Karp [45] can be used to obtain a running time of $O(m\sqrt{n})$. An algorithm of complexity $O(n^\omega)$, where ω is the matrix multiplication constant, was given by Mucha and Sankowski [74]. In graphs where d is a power of 2, the following simple idea, due to Gabow and Kariv [33], leads to an algorithm with $O(m)$ running time. First, compute an Euler tour of the graph (in time $O(m)$) and then follow this tour in an arbitrary direction. Exactly half the edges will go from left to right; these form a regular bipartite graph of degree $d/2$. The total running time $T(m)$ thus follows the recurrence $T(m) = O(m) + T(m/2)$ which yields $T(m) = O(m)$. Extending this idea to the general case proved quite hard, and after a series of improvements (e.g. by Cole and Hopcroft [22], and then by Schrijver [80] to $O(md)$), Cole, Ost, and Schirra [23] gave an $O(m)$ algorithm for the case of general d.

The main interest of Cole, Ost, and Schirra was in edge coloring of general bipartite graphs of maximum degree d, where finding perfect matchings in regular bipartite graphs is an important subroutine. Finding perfect matchings in regular bipartite graphs is also closely related to the problem of finding a Birkhoff von Neumann decomposition of a doubly stochastic matrix [17, 91].

In this chapter we present two algorithms for finding a perfect matching in a regular bipartite graph in *sublinear time*.

For sub-linear (in m) running time algorithms, the exact data model is important. In this paper, as well as in the sub-linear running time algorithms mentioned above, we assume that the graph is presented in the adjacency array format, i.e., for each vertex, its d neighbors are stored in an array. This is the most natural input data structure for our problem. For simple graphs or multigraphs with edge multiplicities bounded above by $\gamma d, \gamma \in (0, 1)$ our algorithms will not make any ordering assumptions within an adjacency array. However, the data layout will be important for multigraphs without any assumptions on edge multiplicities.

2.1 Uniform Sampling for Perfect Matchings

We now present an algorithm for finding a perfect matching in a regular bipartite graph that runs in time $O(\min\{m, \frac{n^{2.5}\ln n}{d}\})$. It is easy to see that this minimum can never be larger than $O(n^{1.75}\sqrt{\ln n})$. This is a significant improvement over the running time of Cole, Ost, and Schirra when the bipartite graph is relatively dense. We first prove (Theorem 1 in section 2.2) that if we sample the edges of a regular bipartite graph independently and uniformly at rate $p = O(\frac{n\ln n}{d^2})$, then the resulting graph has a perfect matching with high probability. The resulting graph has $O(mp)$ edges in expectation, and running the bipartite matching algorithm of Hopcroft and Karp gives an expected running time of $O(\frac{n^{2.5}\ln n}{d})$. Since we know this running time in advance, we can choose the better of m and $\frac{n^{2.5}\ln n}{d}$ in advance. It is worth noting that uniform sampling can easily be implemented in $O(1)$ time per sampled edge assuming that the data is given in adjacency list format, with each list stored in an array, and assuming that $\log n$ bit random numbers can be generated in one time step[1].

We believe that our sampling result is also independently interesting as a combinatorial fact. The proof of our sampling theorem relies on a sequential decomposition procedure that creates a vertex-disjoint collection of subgraphs, each subgraph containing many perfect matchings on its underlying vertex set. We then show that if we uniformly sample edges in each decomposed subgraph at a suitably chosen rate, with high probability at least one perfect matching survives in each decomposed subgraph. This is established by using Karger's sampling theorem [52, 51] in each subgraph. An effective use of Karger's sampling theorem requires the min-cuts to be large, a property that is not necessarily true in the original graph. For instance, G could be a union of two disjoint d-regular bipartite graphs, in which case the min-cut is 0; non-pathological examples are also easy to obtain. However, our serial decomposition procedure ensures that the min-cuts are large in each decomposed subgraph. We then establish a 1-1 correspondence between possible Hall's theorem counter-examples in each subgraph and cuts of comparable size in that subgraph. Since Karger's sampling theorem is based on counting cuts of a certain size, this coupling allows us to claim (with high probability) that no possible counter-example to Hall's theorem exists in the sampled graph. On a related note, Benczúr [15] presented another sampling algorithm which generates $O(n\ln n)$ edges that approximate *all cuts*; however this sampling algorithm, as well as recent improvements [84, 83] take $\tilde{\Omega}(m)$ time to generate the sampled graph. Hence these approaches do not directly help in improving upon the already known $O(m)$ running time for finding perfect matchings in d-regular bipartite graphs.

The sampling rate we provide may seem counter-intuitive; a superficial analogy with Karger's sampling theorem or Benczúr's work might suggest that sampling a total of $O(n\ln n)$ edges should suffice. We show (Theorem 8, section 2.4) that this is not the case. In particular, we present a family of graphs where uniform sampling at rate $o(\frac{n}{d^2\ln n})$ results in a vanishingly low probability that the sampled subgraph has a perfect matching. Thus, our sampling rate is tight up to factors of $O(\ln^2 n)$. This lower bound suggests two promising directions for further research: designing an efficiently implementable non-uniform sampling scheme, and designing an algorithm that runs faster than Hopcroft-Karp's algorithm for near-regular bipartite graphs (since the degree of each vertex in

[1] Even if we assume that only one random bit can be generated in one time step, the running time of our algorithm remains unaltered since the Hopcroft-Karp algorithm incurs an overhead of $\sqrt{n}$ per sampled edge anyway.

the sampled subgraph will be concentrated around the expectation).

2.2 Uniform Sampling for Perfect Matchings: An Upper Bound

In this section, we will establish our main sampling theorem stated below. We will then show in Section 2.3 that this theorem immediately yields an $O(n^{1.75}\sqrt{\ln n})$ time algorithm for finding a perfect matching in regular bipartite graphs.

Theorem 1 *There exists a constant c such that given a d-regular bipartite graph $G(U, V, E)$, a subgraph G' of G generated by sampling the edges in G uniformly at random with probability $p = \frac{cn \ln n}{d^2}$ contains a perfect matching with high probability.*

Our proof is based on a decomposition procedure that partitions the given graph into a vertex-disjoint collection of subgraphs such that (i) the minimum cut in each subgraph is large, and (ii) each subgraph contains $\Omega(d)$ perfect matchings on its vertices. We then show that for a suitable choice of sampling rate, w.h.p. at least one perfect matching survives in each subgraph. The union of these perfect matchings then gives us a perfect matching in the original graph. We emphasize here that the decomposition procedure is merely an artifact for our proof technique. Note that the theorem is trivially true when $d \leq \sqrt{n \log n}$. So in what follows we assume that $d > \sqrt{n \log n}$.

2.2.1 Hall's Theorem Witness Sets

Let $G(U, V, E)$ be a bipartite graph. We denote by $V(G)$ the vertex set of G. For any set $S \subseteq V(G)$, let $\delta_G(S)$ denote the set of edges crossing the boundary of S in G. Also, for any set $S \subseteq V(G)$, we denote by $\Gamma_G(S)$ the set of vertices that are adjacent to vertices in S.

A pair (A, B) with $A \subseteq U$ and $B \subseteq V$ is said to be a *relevant pair* to Hall's theorem if $|A| > |B|$. Given a relevant pair (A, B), we denote by $E(A, B)$ the set of edges in $E \cap (A \times (V \setminus B))$. We refer to the set $E(A, B)$ as a *witness* edge set if (A, B) is a relevant pair. Also, for any two sets $A, A' \subseteq U$ we denote by $A \oplus A'$ the set $(A \setminus A') \cup (A' \setminus A)$. In what follows we will be using Hall's theorem, which we state here for convenience of the reader:

Theorem 2 *(Hall's theorem, cf. [18]) A bipartite graph $G(U, V, E)$ contains a matching that includes every vertex in U iff $|\Gamma_G(S)| \geq |S|$ for all $S \subset U$.*

Note that if $|U| = |V|$, then any matching that includes every vertex in U is a also a perfect matching in G. Since $|U| = |V|$ in a d-regular graph, by Hall's theorem, to prove Theorem 1 it suffices to show that w.h.p. in the sampled graph G', at least one edge is chosen from each witness set. We will focus on a sub-class of relevant pairs, referred to as minimal relevant pairs. A relevant pair (A, B) is *minimal* if there does not exist another relevant pair (A', B') with $A' \subset A$ and $E(A', B') \subseteq E(A, B)$. A witness edge set corresponding to a minimal relevant pair is called a *minimal witness set*, respectively. If a graph G has a perfect matching, every minimal witness set must be non-empty. It also follows from Hall's theorem that any balanced subgraph of G that

includes at least one edge from every minimal witness set must have a perfect matching. We refer to a bipartite graph as balanced if it contains the same number of vertices in each side.

A key idea underlying our proof is a mapping from minimal witness sets in G to *distinct* cuts in G. In particular, we will map each minimal witness set $E(A, B)$ to the cut $\delta_G(A \cup B)$. The theorem below shows that this is a one-to-one mapping.

Theorem 3 *Let $G(U, V, E)$ be a bipartite graph that has at least one perfect matching. If (A, B) and (A', B') are minimal relevant pairs in G with $E(A, B) \neq E(A', B')$, then $\delta_G(A \cup B) \neq \delta_G(A' \cup B')$.*

Proof: Assume by way of contradiction that there exist minimal relevant pairs (A, B) and (A', B') in G with $E(A, B) \neq E(A', B')$ but $\delta_G(A \cup B) = \delta_G(A' \cup B')$. Then the following conditions must be satisfied for any edge $(u, v) \in E$:

A1. If $u \in A \oplus A'$ then $v \in B \oplus B'$. To see this, assume w.l.o.g. that $u \in A \setminus A'$, and then note that if $v \in B \cap B'$, then $(u, v) \in \delta_G(A' \cup B')$ but $(u, v) \notin \delta_G(A \cup B)$, which is a contradiction. Similarly, if $v \in V \setminus (B \cup B')$, then $(u, v) \in \delta_G(A \cup B)$ but $(u, v) \notin \delta_G(A' \cup B')$, which is again a contradiction.

A2. If $u \in (A \cap A')$ then $v \notin B \oplus B'$. To see this, w.l.o.g. assume that $v \in (B \setminus B')$. Then $(u, v) \in \delta_G(A' \cup B')$ but $(u, v) \notin \delta_G(A \cup B)$. This is a contradiction.

In what follows, we slightly abuse the notation and given any (not necessarily relevant) pair (C, D) with $C \subseteq U$ and $D \subseteq V$, we denote by $E(C, D)$ the set of edges in $E \cap (C \times (V \setminus D))$. As an immediate corollary of the properties A1 and A2, we now obtain the following containment results:

B1. $E(A \setminus A', B \setminus B') \subseteq E(A, B)$. This follows directly from property A1 above.

B2. $E(A \cap A', B \cap B') \subseteq E(A, B)$. This follows directly from property A2 above.

We now consider three possible cases based on the relationship between A and A', and establish a contradiction for each case.

Case 1: $A \cap A' = \emptyset$. By property A1, if $u \in A \cup A'$ then $v \in B \cup B'$. In other words, there are no edges from $A \cup A'$ to vertices outside $B \cup B'$. Since $|A \cup A'| = |A| + |A'| > |B| + |B'|$, this contradicts our assumption that G has at least one perfect matching.

Case 2: $A = A'$. For any edge (u, v) with $u \in A$, property A2 shows that $v \notin B \oplus B'$. Then $E(A, B) = E(A', B')$. A contradiction.

Case 3: $A \cap A' \neq \emptyset$ and $A \neq A'$. Assume w.l.o.g. that $A \setminus A' \neq \emptyset$. Since $|A| > |B|$, it must be that either $|A \setminus A'| > |B \setminus B'|$ or $|A \cap A'| > |B \cap B'|$. If $|A \setminus A'| > |B \setminus B'|$, then $(A \setminus A', B \setminus B')$ is a relevant pair, and by B1, it contradicts the fact that (A, B) is a minimal relevant pair. If $|A \cap A'| > |B \cap B'|$, then $(A \cap A', B \cap B')$ is a relevant pair set, and by B2, it contradicts the fact that (A, B) is a minimal relevant pair.

■

2.2.2 A Decomposition Procedure

Given a d-regular bipartite graph on n vertices, we will first show that it can be partitioned into $k = O(n/d)$ vertex disjoint graphs $G_1(U_1, V_1, E_1), G_2(U_2, V_2, E_2), ..., G_k(U_k, V_k, E_k)$ such that each graph G_i satisfies the following properties:

P1. the size of a minimum cut in $G_i(U_i, V_i, E_i)$ is strictly greater than $\alpha = \frac{d^2}{4n}$.

P2. $|\delta_G(U_i \cup V_i)| \leq d/2$ (hence G_i contains at least $d/2$ edge-disjoint perfect matchings).

The decomposition procedure is as follows. Initialize $H_1 = G$, and set $i = 1$.

1. Find a smallest non-empty proper subset $X_i \subseteq V(H_i)$ such that $|\delta_{H_i}(X_i)| \leq 2\alpha$. Let M_i denote the number of edges in the cut $\delta_{H_i}(X_i)$. If no such set X_i exists, we define G_i to be the graph H_i, and terminate the decomposition procedure.

2. Define G_i to be the subgraph of H_i induced by the vertices in X_i, i.e. $X_i = U_i \cup V_i = V(G_i)$. Also, define H_{i+1} to be the graph H_i with vertices from X_i removed.

3. Increment i, and go to step (1).

Note that if the minimum cut of G is greater than 2α, then the procedure terminates after the first step, and the decomposition trivially satisfies both P1 and P2. So we focus below on the case when step (2) is executed at least once.

We now prove the following properties of the decomposition procedure.

Theorem 4 *The decomposition procedure outlined above satisfies properties P1 and P2.*

Proof: We start by proving that property P1 is satisfied. Suppose that there exists a cut $(C, V(G_i) \setminus C)$ in G_i of value at most α, i.e. $|\delta_{G_i}(C)| \leq \alpha$ (note that one could have $C \cap U \neq \emptyset$ and $C \cap V \neq \emptyset$). Let $D = V(G_i) \setminus C$. We have $|\delta_{H_i}(C) \setminus \delta_{G_i}(C)| + |\delta_{H_i}(D) \setminus \delta_{G_i}(D)| \leq 2\alpha$ by the choice of X_i in (1). Suppose without loss of generality that $|\delta_{H_i}(C) \setminus \delta_{G_i}(C)| \leq \alpha$. Then $|\delta_{H_i}(C)| \leq 2\alpha$ and $C \subset X_i$, which contradicts the choice of X_i as the smallest cut of value at most 2α in step (1) of the procedure.

It remains to show that $|\delta_G(U_i \cup V_i)| \leq d/2$ for all i. In order to establish this property, it suffices to show that $\sum_{i=1}^{k} M_i \leq d/2$ (recall that $M_i = |\delta_{H_i}(X_i)|$).

We prove the following statements by induction on k, the number of times step (2) in the decomposition procedure above has been executed thus far.

1. $|V(G_k)| = |U_k \cup V_k| \geq 2d$;

2. $\sum_{i=1}^{k} M_i \leq d/2$;

3. $k + 1 \leq n/d$.

Base: $k = 1$**.** Since $2\alpha = \frac{d^2}{2n} \leq d/2$, we have $M_1 \leq d/2$, which establishes (2). We now prove (1), i.e. show that $G_1(U_1, V_1, E_1)$ has at least $2d$ vertices. Consider any vertex $u \in U_1$. Let $\Gamma_{G_1}(u) \subseteq V_1$ be the neighbors of u in G_1. Clearly,

$$|\delta_G(u) \cap \delta_G(X_1)| + \sum_{v \in \Gamma_{G_1}(u)} |\delta_G(v) \cap \delta_G(X_1)| \leq |\delta_G(X_1)| \leq 2\alpha \leq d/2.$$

If all terms are positive then we have

$$d/2 \geq |\delta_G(u) \cap \delta_G(X_1)| + |\Gamma_{G_1}(u)|.$$

This is a contradiction since the right-hand side is d, the number of neighbors of u. So we have $|\delta_G(v) \cap \delta_G(X_1)| = 0$ for some $v \in \Gamma_{G_1}(u)$, implying that all neighbors of v are inside U_1, so $|U_1| \geq d$. A similar argument shows that $|V_1| \geq d$, so $|X_1| \geq 2d$. $|V(H_2)| \geq 2d$. This establishes (3).

Inductive step: $k - 1 \to k$**.** Suppose that the algorithm constructs G_k. Since $k \leq n/d$ by the inductive hypothesis, we have $\sum_{i=1}^{k} M_i \leq (n/d)\,(2\alpha) = (n/d)\left(\frac{d^2}{2n}\right) \leq d/2$, which establishes (2). Consider the cut $(X_k, V(H_k) \setminus X_k)$ of H_k.

Every edge in $\delta_G(X_k)$ has one endpoint in X_k and the other in either $V(H_k) \setminus X_k$ or $V(G_i) = X_i$ for some $i < k$. Thus

$$\delta_G(X_k) \subseteq \delta_{H_k}(X_k) \cup \bigcup_{i=1}^{k-1} \delta_{H_i}(X_i).$$

Thus

$$|\delta_G(X_k)| \leq M_k + \sum_{i=1}^{k-1} M_i.$$

By induction $k \leq n/d$, so we have

$$|\delta_G(X_k)| \leq \sum_{i=1}^{k} M_i \leq (n/d)(2\alpha) \leq (n/d)\frac{d^2}{2n} \leq d/2.$$

An argument similar to the base case can be used to show that $|X_k| \geq 2d$ as well as $|V(H_k) \setminus X_k| \geq 2d$, establishing (1). Since at every decomposition step $j \leq k$ at least $2d$ vertices were removed from the graph, we have $k + 1 \leq n/d$, which establishes (3).

■

2.2.3 Proof of Theorem 1

We now argue that if the graph G' is obtained by uniformly sampling the edges of G with probability $p = \Theta\left(\frac{\ln n}{\alpha}\right)$, then w.h.p. G' contains a perfect matching.

It suffices to show that in each graph G_i obtained in the decomposition procedure, every minimal witness set is hit w.h.p. in the sampled graph (that is, at least one edge in each minimal witness set is chosen in the sampled graph). This ensures that at least one perfect matching survives inside each G_i. A union of these perfect matchings then gives us a perfect matching of G in the sampled graph G'.

Fix a graph $G_i(U_i, V_i, E_i)$. Let (A, B) be a relevant pair in G_i. Using the fact that our starting graph G is d-regular, we observe that $|E(A,B)| \geq d + |E(B,A)|$, and obtain

$$|\delta_G(A \cup B)| \leq 2|E(A,B)| - d.$$

Let m_A, m_B denote the number of edges in G that connect nodes in A and B, respectively, to nodes outside G_i. Then

$$|\delta_{G_i}(A \cup B)| \leq 2|E(A,B)| - d - m_A - m_B.$$

By property P2, since $|\delta_G(U_i \cup V_i)| \leq d/2$, it follows that $|E(A,B) \cap E_i| \geq |E(A,B)| - d/2$. Also, by definition, $|E(A,B) \cap E_i| \geq |E(A,B)| - m_A - m_B$. Combining, we obtain:

$$|\delta_{G_i}(A \cup B)| \leq 2|E(A,B) \cap E_i| - d/2.$$

Thus the set $E(A,B) \cap E_i$ contains at least half as many edges as the the cut $\delta_{G_i}(A \cup B)$. We will now use the following sampling result due to Karger [51]:

Theorem 5 [51] *Let G_i be an undirected graph on at most n vertices, and let κ be the size of a minimum cut in G_i. There exists a positive constant c such that for any $\epsilon \in (0,1)$, if we sample the edges in G_i uniformly with probability at least $p = c\left(\frac{\ln n}{\kappa \epsilon^2}\right)$, then every cut in G_i is preserved to within $(1 \pm \epsilon)$ of its expected value with probability at least $1 - 1/n^{\Omega(1)}$.*

Thus the sampling probability needed to ensure that all cuts are preserved close to their expected value, is inversely related to the size of a minimum cut in the graph. We now use the theorem above to prove that at least one perfect matching survives in each graph G_i when edges are sampled with probability as specified in Theorem 1.

By Property P1, we know that the size of a minimum cut in G_i is at least $\alpha = d^2/4n$. Fix an $\epsilon \in (0,1)$. The theorem above implies that if we sample edges in G_i with probability $p = \Theta\left(\frac{\ln n}{\alpha \epsilon^2}\right)$, then for every relevant pair (A,B), w.h.p. the sampled graph contains $(1 \pm \epsilon)p|\delta_{G_i}(A \cup B)| = \Omega(\ln n)$ edges from the set $\delta_{G_i}(A \cup B)$.

Note that the set $\delta_{G_i}(A \cup B)$ is not a Hall's theorem witness edge set. However, by Theorem 3, we know that for every left (right) minimal witness edge set $E(A,B) \cap E_i$, we can associate a distinct cut, namely $\delta_{G_i}(A \cup B)$, of size at most twice $|E(A,B) \cap E_i|$. We now show that this correspondence can be used to directly adapt Karger's proof of Theorem 5 to claim that every witness edge set in G_i is preserved to within $(1 \pm \epsilon)$ of its expected value. We remind the reader that the proof of Karger's theorem is based on an application of union bound over all cuts in the graph. In particular,

it is shown that the number of cuts of size at most β times the minimum cut size is bounded by $n^{2\beta}$. Then, for the sampling rate given in Theorem 5, Chernoff bounds are used to claim that the probability that a cut of size β times the minimum cut deviates by $(1 \pm \epsilon)$ from its expected value is at most $1/n^{\Omega(\beta)}$. The theorem follows by combining these two facts.

Within any piece of the decomposition, let c_i be the number of cuts of size i and let w_i be the number of minimal witness sets of size i. We know by the correspondence argument above that every Hall's theorem minimal witness set of size i corresponds to a cut of size at most $2i$, and at most one minimal witness set corresponds to the same cut.

Now, given a sampling probability p, the probability that none of the edges in some minimal witness set are sampled is at most $\sum_i w_i(1-p)^i$, which is at most $\sum_i c_i(1-p)^{i/2}$. Therefore the probability that there is no matching in this piece can be at most twice the expression used in Karger's theorem to bound the probability that there exists a cut from which no edge is sampled when the sampling rate is q, where $1-q = (1-p)^{1/2}$, or $p = 2q - q^2$. Hence, it is sufficient to use a sampling rate which is twice that required by Karger's sampling theorem to conclude that a perfect matching survives with probability at least $1 - 1/n^{\Omega(1)}$ in any given piece of the decomposition. The union bound over all pieces of decomposition can be handled by increasing the constant in the sampling probability.

Even though we don't use it in this paper, the following remark is interesting and is worth making explicitly. The remark follows from the additional observation that Karger's proof [51] of theorem 5 uses Chernoff bounds for each cut, and these bounds remain the same if we use minimal witness sets which are at least half the size of the corresponding cuts, and then sample with twice the probability.

Remark 6 *There exists a positive constant c' such that for any $\epsilon \in (0,1)$, if we sample the edges in G uniformly with probability at least $p = c'\left(\frac{\ln n}{\alpha\epsilon^2}\right)$, then every minimal witness set in every piece G_i is preserved to within $(1 \pm \epsilon)$ of its expected value with probability at least $1 - 1/n^{\Omega(1)}$. Here $\alpha = d^2/(4n)$, as defined before.*

Putting everything together, the sampled graph G' will have a perfect matching w.h.p. as long as we sample the edges with probability $p > \frac{c \ln n}{\alpha}$ for a sufficiently large constant c, thus completing the proof of theorem 1. We have made no attempt to optimize the constants in this proof (an upper bound of $\frac{8 \ln n}{\alpha}$ follows from the reasoning above). In fact, in an implementation, we can use geometrically increasing sampling rates until either the sampled graph has a perfect matching, or the sampling rate becomes so large that the expected running time of the Hopcroft and Karp [45] algorithm is $\Omega(m)$.

2.3 A Faster Algorithm for Perfect Matchings in Regular Bipartite Graphs

We now show that the sampling theorem from the preceding section can be used to obtain a faster randomized algorithm for finding perfect matchings in d-regular bipartite graphs.

Theorem 7 *There exists an* $O(\min\{m, \frac{n^{2.5}\ln n}{d}\})$ *expected time algorithm for finding a perfect matching in a* d*-regular bipartite graph with* $2n$ *vertices and* $m = nd$ *edges.*

Proof: Let G be a d-regular bipartite graph with $2n$ vertices and $m = nd$ edges. If $d \leq n^{3/4}\sqrt{\ln n}$, we use the $O(m)$ time algorithm of Cole, Ost, and Schirra [23] for finding a perfect matching in a d-regular bipartite graph. It is easy to see that $m \leq \frac{n^{2.5}\ln n}{d}$ in this case.

Otherwise, we sample the edges in G at a rate of $p = \frac{cn\ln n}{d^2}$ for some suitably large constant c ($c = 32$ suffices by the reasoning from the previous section), and by Theorem 1, the sampled graph G' contains a perfect matching w.h.p. The expected number of edges, say m', in the sampled graph G' is $O(\frac{n^2\ln n}{d})$. We can now use the algorithm of Hopcroft and Karp [45] to find a maximum matching in the bipartite graph G' in expected time $O(m'\sqrt{n})$. The sampling is then repeated if no perfect matching exists in G'. This takes $O(\frac{n^{2.5}\ln n}{d})$ expected running time. Hence, the algorithm takes $O(\min\{m, \frac{n^{2.5}\ln n}{d}\})$ expected time. ■

Note that by aborting the computation whenever the number of sampled edges is more than twice the expected value, the above algorithm can be easily converted to a Monte-Carlo algorithm with a worst-case running time of $O(\min\{m, \frac{n^{2.5}\ln n}{d}\})$ and a probability of success $= 1 - o(1)$. Finally, it is easy to verify that the stated running time never exceeds $O(n^{1.75}\sqrt{\ln n})$.

2.4 Uniform Sampling for Perfect Matchings: A Lower Bound

We now present a construction that shows that the uniform sampling rate of Theorem 1 is optimal to within a factor of $O(\ln^2 n)$. As before, G' denotes the graph obtained by sampling the edges of a graph G uniformly with probability p.

Theorem 8 *Let* $d(n)$ *be a non-decreasing positive integer valued function such that for some fixed integer* n_0*, it always satisfies one of the following two conditions for all* $n \geq n_0$*: (a)* $d(n) \leq \sqrt{n/\ln n}$*, or (b)* $\sqrt{n/\ln n} < d(n) \leq n/\ln n$*. Then there exists a family of* $d(n)$*-regular bipartite graphs* G_n *with* $2n + o(n)$ *vertices such that the probability that the graph* G'_n*, obtained by sampling edges of* G_n *with probability* p*, has a perfect matching goes to zero faster than any inverse polynomial function in* n *if* $p = o(1)$ *when* $d(n)$ *satisfies condition (a) above, and if*

$$p = o\left(\frac{n}{(d(n))^2 \ln n}\right)$$

when $d(n)$ *satisfies condition (b) above.*

Proof: Note that the theorem asserts that essentially no sampling can be done when $d(n) \leq \sqrt{n/\ln n}$. We shall omit the dependence on n in $d(n)$ to simplify notation.

Define $H^{(k)} = (U, V, E)$, $0 \leq k \leq d$, to be a bipartite graph with $|U| = |V| = d$ such that k vertices in each of U and V have degree $(d-1)$ and the remaining vertices have degree d. We will call the vertices of degree $(d-1)$ *deficient*. Clearly, for any $0 \leq k \leq d$, the graph $H^{(k)}$ exists: starting with a d-regular bipartite graph on $2d$ vertices, we can remove an arbitrary subset of k edges that

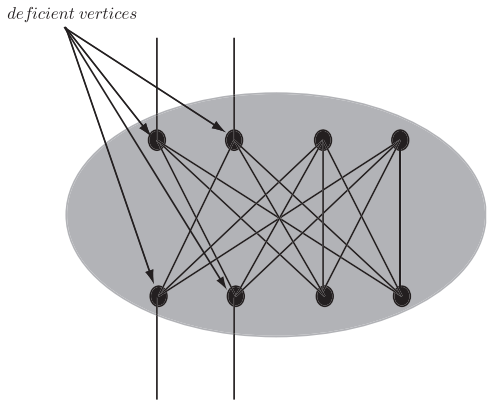


Figure 2.1: Graph $H^{(k)}$ for $k = 2$ and $d = 4$.

belong to a perfect matching in the graph ($H^{(k)}$ with $k = 2$ and $d = 4$ is shown in Fig. 2.1). In the following construction, we will use copies of $H^{(k)}$ as building blocks to create our final instance. In doing so, only the set of deficient vertices in a copy of $H^{(k)}$ will be connected to (deficient) vertices in other copies in our construction.

We now define a d-regular bipartite graph G_n. Let $\gamma = \left\lceil \frac{d^2 \ln n}{n} \right\rceil$ (note that $\gamma \leq d$ since $d \leq n/\ln n$). We choose $W = \left\lceil \frac{d}{\gamma} \right\rceil$, $k_j = \gamma$ for $1 \leq j < W$, and $k_W = d - \gamma(W-1) \leq \gamma$. We also define $K(n) = \lceil \ln n \rceil$ if $d(n) \geq \sqrt{n/\ln n}$ and $K(n) = \lceil \frac{n}{d^2} \rceil$ otherwise.

The graph G_n consists of $K(n) \cdot W$ copies of $H^{(k)}$ that we index as $\{H_{i,j}\}_{1 \leq i \leq K(n), 1 \leq j \leq W}$. The subgraph $H_{i,j}$ is a copy of $H^{(k_j)}$, where k_j is as defined above. Note that the sum of the number of deficient vertices over each of the parts of $H_{i,j}, 1 \leq j \leq W$, equals d for all fixed i. Moreover, the number of deficient vertices in $H_{i,j}$ is the same for all i when j is held fixed.

We now introduce two distinguished vertices u and v and add additional edges as follows:

1. For every $1 \leq i < K(n)$ and for every $1 \leq j \leq W$, all deficient vertices in part V of $H_{i,j}$ are matched to the deficient vertices in part U of $H_{i+1,j}$ (that is, we insert an arbitrary matching between these two sets of vertices);

2. All deficient vertices in part U of $H_{1,j}$ for $1 \leq j \leq W$ are connected to u;

3. All deficient vertices in part V of $H_{K(n),j}$ for $1 \leq j \leq W$ are connected to v.

Essentially, we are connecting the graphs $H_{i,j}$ for fixed j in series via their deficient vertices, and then connecting the left ends of these chains to the distinguished vertex u and the right ends of the chains to the distinguished vertex v. The construction is illustrated in Fig. 2.2.

We note that the graph G_n constructed as described above is a d-regular bipartite graph with $2dK(n)W + 2 = 2n + o(n)$ vertices.

Consider the sampled graph G'_n. Suppose G'_n has a perfect matching M. In the matching M, if u is matched to a vertex in part U of $H'_{1,j}$ for some $1 \leq j \leq W$, then there must be a vertex in part V of $H'_{1,j}$ that is matched to a vertex in part U of $H'_{2,j}$. Proceeding in the same

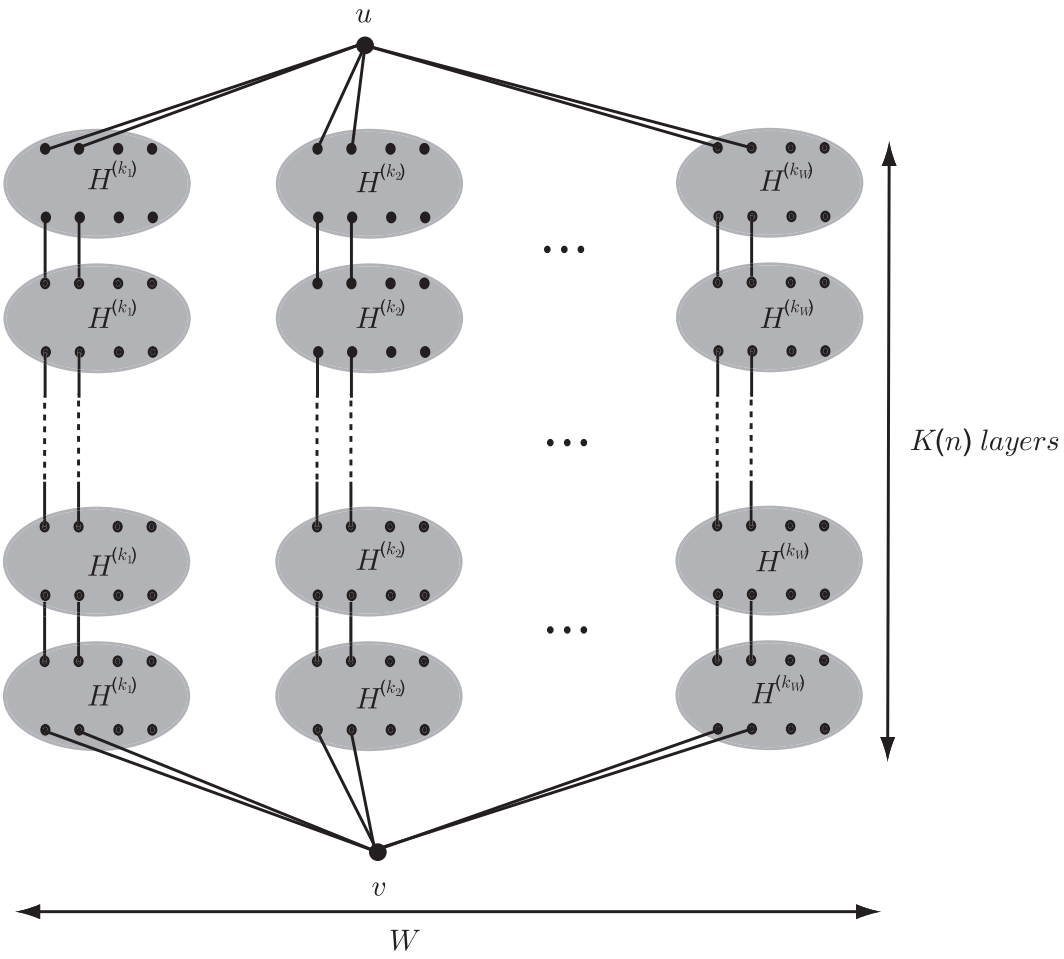


Figure 2.2: Illustration of the family of graphs that yields the lower bound.

way, one concludes that for every $i, 1 \leq i < K(n)$ there must be a vertex in part V of $H'_{i,j}$ that is matched to a vertex in part U of $H'_{i+1,j}$. Finally, vertex v must be matched to a vertex in part V of $H'_{K(n),j}$. This implies that the sampled graph G'_n can have a perfect matching only if at least one edge survives in G'_n between every pair of adjacent elements in the sequence below: $u \to H_{1,j} \to H_{2,j} \to \ldots \to H_{K(n)-1,j} \to H_{K(n),j} \to v$.

Now suppose that we sample edges uniformly with probability p. It follows from the construction of G_n that for any fixed j, the probability that at least one edge survives between every pair of adjacent elements in the sequence $u \to H_{1,j} \to H_{2,j} \to \ldots \to H_{K(n)-1,j} \to H_{K(n),j} \to v$ is equal to

$$\left(1 - (1-p)^{k_j}\right)^{K(n)+1} \leq (pk_j)^{K(n)+1}.$$

Hence, the probability that at least one such path survives in G'_n is at most

$$W\left(p \max_{1 \leq j \leq W} k_j\right)^{K(n)+1}$$

by the union bound.

When $d(n) \leq \sqrt{n/\ln n}$, we have $\gamma = 1$, $W = d$, $k_j = 1$ and $K(n) = \lceil n/d^2 \rceil$. So the bound transforms to

$$Wp^{K(n)+1} = dp^{\lceil n/d^2 \rceil + 1}, \tag{2.1}$$

which goes to zero faster than any inverse polynomial function in n when $p = o(1)$ since $K(n) = \lceil n/d^2 \rceil = \Omega(\ln n)$.

When $d \geq \sqrt{n/\ln n}$, we have $k_j \leq \gamma$ where $\gamma = \left\lceil \frac{d^2 \ln n}{n} \right\rceil$, $W = \left\lceil \frac{d}{\gamma} \right\rceil$ and $K(n) = \lceil \ln n \rceil$. Hence, the bound becomes

$$W(p\gamma)^{K(n)+1} = \left\lceil \frac{d}{\gamma} \right\rceil (p\gamma)^{\lceil \ln n \rceil + 1}, \tag{2.2}$$

which goes to zero faster than any inverse polynomial function in n when $p = o\left(\frac{n}{d^2 \ln n}\right)$. This completes the proof of the theorem.

■

The construction given in Theorem 8 shows that the sampling upper bound for preserving a perfect matching proved in Theorem 1 is tight up to a factor of $O(\ln^2 n)$.

2.5 Matchings via Random Walks

In section 2.1, we gave a sampling-based algorithm that computes a perfect matching in d-regular bipartite graphs in $O(\min\{m, \frac{n^{2.5}\log n}{d}\})$ expected time, an expression that is bounded by $\tilde{O}(n^{1.75})$. That algorithm uses uniform sampling to reduce the number of edges in the input graph while preserving a perfect matching, and then runs the Hopcroft-Karp algorithm on the sampled graph. We also gave a lower bound of $\tilde{\Omega}\left(\min\{nd, \frac{n^2}{d}\}\right)$ on the running time of an algorithm that uses non-adaptive uniform sampling to reduce the number of edges in the graph as the first step. Also, in Chapter 7 we match this lower bound by using a two stage sampling scheme and a specialized analysis of the runtime of the Hopcroft-Karp algorithm on the sampled graph to obtain a runtime of $\tilde{O}\left(\min\{nd, \frac{n^2}{d}\}\right)$.

Given a partial matching in an undirected graph, an *augmenting path* is a path which starts and ends at an unmatched vertex, and alternately contains edges that are outside and inside the partial matching. Many of the algorithms mentioned above work by repeatedly finding augmenting paths.

Our main result in the rest of this chapter is the following theorem.

Theorem 9 *There exists a randomized algorithm for finding a perfect matching in a d-regular bipartite graph $G = (P, Q, E)$ given in adjacency array representation, and takes time $O(n \log n)$ time both in expectation as well as with high probability.*

The algorithm is very simple: the matching is constructed by performing one augmentation at a time, and new augmenting paths are found by performing an *alternating random walk* with respect to the current matching. The alternating random walk on G, defined in Section 2.6, can be viewed as a random walk on a modified graph that encodes the current matching. The random walk approach may still be viewed as repeatedly drawing a uniform sample from the adjacency array of some vertex v; however this vertex v is now chosen adaptively, thus allowing us to bypass the $\tilde{\Omega}\left(\min\{nd, \frac{n^2}{d}\}\right)$ lower bound on non-adaptive uniform sampling established in section 2.1. Somewhat surprisingly, we show that the total time taken by these random augmentations can be bounded by $O(n \log n)$ in expectation, only slightly worse than the $\Omega(n)$ time needed to simply output a perfect matching. The proof involves analyzing the hitting time of the sink node in the random walk. It should be noted here that, of course, the $O(m)$ algorithm of Cole-Ost-Schirra is faster than our algorithm when $d = o(\log n)$.

In section 2.8 we establish that randomization is crucial to obtaining an $o(nd)$ time algorithm, thus showing that the algorithm of [23] is asymptotically optimal in the class of deterministic algorithms. In section 2.9, we show that no *randomized* algorithm can achieve *high probability* runtime better than $O(n \log n)$ for the case of d-regular bipartite multigraphs.

Our techniques also extend to the problem of finding a perfect matching in the support of a doubly-stochastic matrix, as well as to efficiently compute the Birkhoff-von-Neumann decomposition of a doubly stochastic matrix. The details are given in section 2.7.1.

Finally, we note that an application of Yao's min-max theorem (see, for instance, [72]) to Theorem 9 immediately yields the following corollary:

Corollary 10 *For any distribution on regular bipartite graphs with $2n$ nodes, there exists a deterministic algorithm that runs in average time $O(n \log n)$ on graphs drawn from this distribution.*

A similar corollary also follows for doubly stochastic matrices.

2.6 Matchings in d-Regular Bipartite Graphs

2.6.1 The Basic Algorithm

Let $G = (P, Q, E)$ denote the input d-regular graph and let M be a partial matching in G. We first describe the *alternating random walk* on G with respect to M. We assume that the algorithm has access to the function SAMPLE-OUT-EDGE that takes a vertex $u \in P$ and returns a uniformly random unmatched edge going out of u. The implementation and runtime of SAMPLE-OUT-EDGE depend on the representation of the graph. It is assumed in Theorem 9 and in this section that the graph G does not have parallel edges and is represented in adjacency array format, in which case SAMPLE-OUT-EDGE can be implemented to run in expected constant time (throughout the paper we assume the model in which a random number in the range $1 : n$ can be generated in $O(1)$ time). In Theorem 13, however, a preprocessing step will be required to convert the matrix to an augmented binary search tree, in which case SAMPLE-OUT-EDGE can be implemented to run in $O(\log n)$ time.

The alternating random walk starts at a uniformly random unmatched vertex $u_0 \in P$ and proceeds as follows:

1. Set $v :=$SAMPLE-OUT-EDGE(u_j);
2. If v is matched, set $u_{j+1} := M(v)$, otherwise terminate.

Note that an augmenting path with respect to M can be obtained from the sequence of steps taken by the alternating random walk by removing possible loops.

We now state a basic version of our algorithm:

Algorithm 1

Input: A d-regular bipartite graph $G = (P, Q, E)$ in adjacency array format.

Output: A perfect matching of G.

1. Set $j := 0$, $M_0 := \emptyset$.

2. Run the alternating random walk starting from a random unmatched vertex in P until it hits an unmatched vertex in Q.

3. Denote the augmenting path obtained by removing possible loops from the sequence of steps taken by the walk by p. Set $M_{j+1} := M_j \Delta p$.

4. Set $j := j + 1$ and go to step 2.

Here for two sets of edges $A, B \subseteq E$ we use the notation $A \Delta B$ to denote the symmetric difference of A and B. In particular, if M is a matching and p is an augmenting path with respect to M, then $M \Delta p$ is the set of edges obtained by augmenting M with p.

We prove in the next section that this algorithm takes $O(n \log n)$ time in expectation. The high probability result is obtained in section 2.6.3 by performing appropriately truncated random walks in step 2 instead of a single untruncated walk.

2.6.2 Expected Running Time Analysis

The core of our analysis is the following lemma, which bounds the time that it takes an alternating random walk in G with respect to a partial matching M that leaves $2k$ vertices unmatched to reach an unmatched vertex.

Lemma 11 *Let $G = (P, Q, E)$ be a d-regular bipartite graph and let M be a partial matching that leaves $2k$ vertices unmatched. Then the expected number of steps before the alternating random walk in G reaches an unmatched vertex is at most $4 + n/k$.*

Proof:

It will be convenient to use the auxiliary notion of a *matching graph* $H(G, M)$ which will allow us to view alternating random walks in G with respect to M as random walks in $H(G, M)$ starting from a special source node s and hitting a special sink node t. We then get the result by bounding the hitting time from s to t in $H(G, M)$.

The matching graph corresponding to the matching M is defined to be the directed graph H obtained by transforming G as follows:

1. Orient edges of G from P to Q;

2. Add a vertex s connected by d parallel edges to each unmatched node in P, directed out of s;

3. Add a vertex t connected by d parallel edges to each unmatched node in Q, directed into t;

4. Contract each pair $(u, v) \in M$ into a supernode.

The graph H has $n + k + 2$ nodes. Note that for every vertex $v \in H$, $v \neq s, t$ the in-degree of v is equal to its out-degree, the out-degree of s equals dk, as is the in-degree of t. Also, any path from s to t in H gives an augmenting path in G with respect to M. We now concentrate on finding a path from s to t in H.

Construct the graph H^* by identifying s with t in H and adding a self-loop at every vertex (and thus increasing all degrees by 1). Denote the vertex that corresponds to s and t by s^*. Note that H^* is a balanced directed graph, i.e. the out-degree of every vertex is equal to its in-degree. The out-degree of every vertex except s^* is at most $d + 1$, while the out-degree of s^* is $dk + 1$.

Consider the simple random walk in H^* started at s^* where at each step of the random walk, we choose an outgoing edge uniformly at random. We wish to analyze the expected time for a return visit to s^*; this is precisely the expected time to find an augmenting path. We first observe that since H^* is a balanced directed graph, any strongly connected component must have the same number of edges entering and leaving that component. It follows that every strongly connected component is isolated, and hence the Markov chain induced by the above simple random walk is irreducible. Furthermore, the addition of self-loops ensures that the chain is aperiodic. Denote the set of vertices in the strongly connected component of s^* by C.

By the Fundamental theorem of Markov chains (see, e.g. [88], Theorem 4.1, where this result is referred to as the basic limit theorem of Markov chains), we know that there is a unique stationary distribution, and it is easy to verify that it is given by

$$\pi_u = \frac{\deg(u)}{\sum_{v \in C} \deg(v)}$$

for each $u \in C$. Since the expected return time to s^* is equal to the inverse of the stationary measure of s^*, we get that

$$\frac{1}{\pi_{s^*}} = \frac{\sum_{v \in C} \deg(v)}{\deg(s^*)} = \frac{(n-k)d + 2k(d+1) + dk + 1}{dk + 1} \leq 1 + \frac{(n+k)(d+1)}{dk} \leq 4 + \frac{2n}{k}.$$

■

We can now prove

Theorem 12 *Algorithm 1 finds a matching in a d-regular bipartite graph $G = (P, Q, E)$ in expected time $O(n \log n)$.*

Proof: By Lemma 11 it takes at most $4 + 2n/(n - j)$ expected time to find an augmenting path with respect to partial matching M_j. Hence, the expected runtime of the algorithm is bounded by

$$\sum_{j=0}^{n-1} 4 + 2n/(n-j) = 4n + 2nH_n = O(n \log n),$$

where $H(n) := 1 + 1/2 + 1/3 + \ldots + 1/n$ is the n-th Harmonic number. ■

2.6.3 Truncated Random Walks and High Probability Analysis

In this section we show how Algorithm 1 can be modified by introducing truncated random walks to obtain a running time of $O(n \log n)$ with high probability.

Algorithm 2

Input: A d-regular bipartite graph $G = (P, Q, E)$ in adjacency array format.

Output: A perfect matching of G.

1. Set $j := 0$, $M_0 := \emptyset$.

2. Repeatedly run alternating random walks for $b_j := 2\left(4 + \frac{2n}{n-j}\right)$ steps until a successful run is obtained.

3. Denote the augmenting path obtained by removing possible loops from the sequence of steps taken by the walk by p. Set $M_{j+1} := M_j \Delta p$.

4. Set $j := j + 1$ and go to step 2.

We now analyze the running time of our algorithm, and prove Theorem 9.

Proof of Theorem 9:

We now show that Algorithm 2 takes time $O(n \log n)$ with high probability. First note that by Lemma 11 and Markov's inequality, a truncated alternating random walk in step 2 succeeds with probability at least $1/2$. Let X_j denote the time taken by the j-th augmentation. Let Y_j be independent exponentially distributed with mean $\mu_j := \frac{b_j}{\ln 2}$. Note that

$$\mathbf{Pr}[X_j \geq q b_j] \leq 2^{-q} = \exp\left[-\frac{q b_j \ln 2}{b_j}\right] = \mathbf{Pr}[Y_j \geq q b_j]$$

for all $q > 1$, so

$$\mathbf{Pr}[X_j \geq x] \leq \mathbf{Pr}[Y_j \geq x] \tag{2.3}$$

for all $x > b_j$. We now prove that $Y := \sum_{0 \leq j \leq n-1} Y_j \leq cn \log n$ w.h.p. for a suitably large positive constant c. Denote $\mu := \mathbf{E}[Y]$. By Markov's inequality, for any $t, \delta > 0$

$$\mathbf{Pr}[Y \geq (1+\delta)\mu] \leq \frac{\mathbf{E}[e^{tY}]}{e^{t(1+\delta)\mu}}.$$

Also, for any j, and for $t < 1/\mu_j$, we have

$$\mathbf{E}[e^{tY_j}] = \frac{1}{\mu_j} \int_0^\infty e^{tx} e^{-x/\mu_j} dx = \frac{1}{1 - t\mu_j}.$$

The two expressions above, along with the fact that the Y_j's are independent, combine to give:

$$\mathbf{Pr}[Y \geq (1+\delta)\mu] \leq \frac{e^{-t(1+\delta)\mu}}{\prod_{j=0}^{n-1}(1-t\mu_j)}. \tag{2.4}$$

Observe that μ_{n-1} is the largest of the μ_j's. Assume that $t = \frac{1}{2\mu_{n-1}}$, which implies that $(1-t\mu_j) \geq e^{-t\mu_j \ln 4}$. Plugging this into equation 2.4, we get:

$$\mathbf{Pr}[Y \geq (1+\delta)\mu] \leq e^{-(1+\delta-\ln 4)\mu/(2\mu_{n-1})}. \tag{2.5}$$

Further observe that $\mu = \frac{4}{\ln 2}\sum_{j=0}^{n-1}(1+n/(n-j)) = \frac{4}{\ln 2}n(1+H_n) \geq \mu_{n-1}H(n)$, where $H(n) := 1+1/2+1/3+\ldots+1/n$ is the n-th Harmonic number. Since $H(n) \geq \ln n$, we get our high probability result:

$$\mathbf{Pr}[Y \geq (1+\delta)\mu] \leq n^{-(1+\delta-\ln 4)/2}. \tag{2.6}$$

Since $\mu = O(n \log n)$, this completes the proof of Theorem 9. ■

2.7 Matchings in Doubly-Stochastic Matrices and Regular Bipartite Multigraphs

2.7.1 Doubly-Stochastic Matrices

We now apply techniques of the previous section to the problem of finding a perfect matching in the support of an $n \times n$ doubly stochastic matrix $\mathbf{M}$ with m non-zero entries. A doubly-stochastic matrix can be viewed as a regular graph, possibly with parallel edges, and we can thus use the same algorithm and analysis as above, provided that SAMPLE-OUT-EDGE can be implemented efficiently. We start by describing a simple data structure for implementing SAMPLE-OUT-EDGE. For each vertex v, we store all the outgoing edges from v in a balanced binary search tree, augmented so that each node in the search tree also stores the weight of all the edges in its subtree. Since inserts into, deletes from, and random samples from this augmented tree all take time $O(\log n)$, we obtain a running time of $O(n \log^2 n)$ for finding a matching in the support of a doubly stochastic matrix.

Superficially, it might seem that initializing the balanced binary search trees for each vertex takes total time $\Theta(m \log n)$. However, note that there is no natural ordering on the outgoing edges from a vertex, and we can simply superimpose the initial balanced search tree for a vertex on the adjacency array for that vertex, assuming that the underlying keys are in accordance with the (arbitrary) order in which the edges occur in the adjacency array. We have proved

Theorem 13 *Given an $n \times n$ doubly-stochastic matrix M with m non-zero entries, one can find a perfect matching in the support of M in $O(n \log^2 n)$ expected time with $O(m)$ preprocessing time.*

In many applications of Birkhoff von Neumann decompositions (e.g. routing in network switches [20]), we need to find one perfect matching in a single iteration, and then update the weights of the matched

edges. In such applications, each iteration can be implemented in $O(n \log^2 n)$ time (after initial $O(m)$ preprocessing time), improving upon the previous best known bound of $O(mb)$ where b is the bit precision.

The complete Birkhoff-von Neumann decomposition can be computed by subtracting an appropriately weighted matching matrix from $\mathbf{M}$ every time a matching is found, thus decreasing the number of nonzero entries of $\mathbf{M}$. Note that the augmented binary search tree can be maintained in $O(\log n)$ time per deletion. This yields

Corollary 14 *For any $k \geq 1$, there exists an $O(m + kn \log^2 n)$ expected time algorithm for finding k distinct matchings (if they exist) in the Birkhoff-von-Neumann decomposition of an $n \times n$ doubly stochastic matrix with m non-zero entries.*

Remark 15 *The $O(\log n)$ bound on the time per sample and update that is achieved via binary search trees can be improved using the results of Hagerup et al[42] on sampling from a changing distribution in constant time under the assumption that the entries of $\mathbf{M}$ are represented using $O(\log n)$ bits. This yields an expected runtime of $O(n \log n)$ for finding one matching in Theorem 13 and $O(m + kn \log n)$ expected time in Corollary 14.*

2.7.2 Regular Bipartite Multigraphs and Edge Coloring

For regular bipartite multigraphs with edge multiplicities at most $d/2$, Algorithm 2 still takes time at most $O(n \log n)$ with high probability, since SAMPLE-OUT-EDGE can be implemented by sampling the adjacency list of the appropriate vertex in G until we find an unmatched edge. Each sample succeeds with probability at least $1/2$ since the matched edge can have multiplicity at most $d/2$. Here, we assume that an edge with multiplicity k occurs k times in the adjacency arrays of its endpoints.

We also note that our algorithm can be implemented to run in $O(n \log n)$ time without any assumptions on multiplicities if the data layout is as follows. For each vertex we have an adjacency array with edges of multiplicity k appearing as contiguous blocks of length k. Also, each element in the adjacency array is augmented with the index of the beginning of the block corresponding to its edge and the index of the end of the block. Assuming this data layout, SAMPLE-OUT-EDGE can be implemented in $O(1)$ expected time regardless of edge multiplicities: it is sufficient to sample locations outside the block corresponding to the currently matched edge.

It remains to note that when the size of the support of the set of edges, which we denote by m_s, is small, then the data representation used in finding a matching in a doubly-stochastic matrix can be used to find a matching in time $O(m_s + n \log^2 n)$. It is interesting to compare this runtime to the result of [23]. The runtime of their matching algorithm is stated as $O(m + n \log^3 d) = O(m)$, but it is easy to see that it can be implemented to run in $O(m_s + n \log^3 d)$ time.

Our algorithm can be used to obtain a simple algorithm for edge-coloring bipartite graphs with maximum degree d in time $O(m \log n)$ (slightly worse than the best known $O(m \log d)$ dependence obtained in [32, 23, 47, 78]). In the first step one reduces the problem to that on a regular graph

with $O(m)$ edges as described in [23] (note that parallel edges may emerge at this point). The lists of neighbors of every vertex of the graph can then be arranged in a data structure that supports sampling and deletion in $O(1)$ amortized time. It then remains to find matchings repeatedly, taking $O(n \log n)$ time per matching. This takes $O(nd \log n) = O(m \log n)$ time overall.

One simple approach to implementing sampling and deletion in $O(1)$ amortized time is to use the data layout outlined above: for each vertex we have an adjacency array with edges of multiplicity k appearing as contiguous blocks of length k. Also, each element in the adjacency array is augmented with the index of the beginning of the block corresponding to its edge and the index of the end of the block. Deletion is performed as follows. When an edge is deleted, its block is marked for deletion (it is sufficient to store a corresponding flag at the beginning of each contiguous block), but is only removed when the number of elements marked for deletion (counting multiplicities) exceeds half of the current size of the adjacency array, in which case the whole array is rearranged, removing the marked elements. Note that until the number of elements in the deleted blocks exceeds half of the current size of the array, sampling can be performed in $O(1)$ expected time. On the other hand, since rearrangement of the array takes linear time and is only performed when at least half of the elements are marked for deletion, the amortized cost of deletion is $O(1)$.

2.8 An $\Omega(nd)$ Lower Bound for Deterministic Algorithms

In this section, we will prove

Theorem 16 *For any $1 \leq d < n/12$, there exists a family of d-regular graphs on which any deterministic algorithm for finding a perfect matching requires $\Omega(nd)$ time.*

We will show that for any positive integer d, any deterministic algorithm to find a perfect matching in a d-regular bipartite graph requires $\Omega(nd)$ probes, even in the adjacency array representation, where the ordering of edges in an array is decided by an adversary. Specifically, for any positive integer d, we construct a family $\mathcal{G}(d)$ of simple d-regular bipartite graphs with $O(d)$ vertices each that we refer to as *canonical* graphs. A canonical bipartite graph $G(P \cup \{t\}, Q \cup \{s\}, E) \in \mathcal{G}(d)$ is defined as follows. The vertex set $P = P_1 \cup P_2$ and $Q = Q_1 \cup Q_2$ where $|P_i| = |Q_i| = 3d$ for $i \in \{1, 2\}$. The vertex s is connected to an arbitrary set of d distinct vertices in P_1 while the vertex t is connected to an arbitrary set of d distinct vertices in Q_2. In addition, G contains a matching M' of size d that connects a subset $Q_1' \subseteq Q_1$ to a subset $P_2' \subseteq P_2$, where $|Q_1'| = |P_2'| = d$. The remaining edges in E connect vertices in P_i to Q_i for $i \in \{1, 2\}$ so as to satisfy the property that the degree of each vertex in G is exactly d. It suffices to show an $\Omega(d^2)$ lower bound for graphs drawn from $\mathcal{G}(d)$ since we can take $\Theta(n/d)$ disjoint copies of canonical graphs to create a d-regular graph on n vertices.

Overview: Let $\mathcal{D}$ be a deterministic algorithm for finding a perfect matching in graphs drawn from $\mathcal{G}(d)$. We will analyze a game between the algorithm $\mathcal{D}$ and an adaptive adversary $\mathcal{A}$ whose goal is to maximize the number of edges that $\mathcal{D}$ needs to examine in order to find a perfect matching. In order to find a perfect matching, the algorithm $\mathcal{D}$ must find an edge in M', since s must be matched

to a vertex in P_1, and thus in turn, some vertex in Q_1 must be matched to a vertex in P_2. We will show that the adversary $\mathcal{A}$ can always force $\mathcal{D}$ to examine $\Omega(d^2)$ edges in G before revealing an edge in M'. The specific graph $G \in \mathcal{G}(d)$ presented to the algorithm depends on the queries made by the algorithm $\mathcal{D}$. The adversary adaptively answers these queries while maintaining at all times the invariant that the partially revealed graph is a subgraph of some graph $G \in \mathcal{G}(d)$. The cost of the algorithm is the number of edge locations probed by it before $\mathcal{A}$ reveals an edge in M' to $\mathcal{D}$.

In what follows, we assume that the adversary reveals s, t and the partition of remaining vertices into P_i, Q_i for $1 \leq i \leq 2$, along with all edges from s to P_1 and all edges from t to Q_2, to the deterministic algorithm $\mathcal{D}$ at the beginning. The algorithm pays no cost for this step.

Queries: Whenever the algorithm $\mathcal{D}$ probes a new location in the adjacency array of some vertex $u \in P \cup Q$, we will equivalently view $\mathcal{D}$ as making a query $\mathcal{Q}(u)$ to the adversary $\mathcal{A}$, in response to which the adversary outputs a vertex v that had not been yet revealed as being adjacent to u.

Subgraphs consistent with canonical graphs: Given a bipartite graph $G'(P \cup \{t\}, Q \cup \{s\}, E')$, we say that a vertex $u \in P \cup Q$ is *free* if its degree in G' is strictly smaller than d. We now identify sufficient conditions for a partially revealed graph to be a subgraph of some canonical graph in $\mathcal{G}(d)$.

Lemma 17 *Let $G_r(P \cup \{t\}, Q \cup \{s\}, E_r)$ be any simple bipartite graph such that*

(a) the vertex s is connected to d distinct vertices in P_1 and the vertex t is connected to d distinct vertices in Q_2,

(b) all other edges in G_r connect a vertex in P_i to a vertex in Q_i for some $i \in \{1, 2\}$,

(c) degree of each vertex in G_r is at most d, and

(d) at least $\frac{5d}{2}$ vertices each in both Q_1 and P_2 have degree strictly less than $\frac{d}{5}$.

Then for any pair u, v of free vertices such that $u \in P_i$ and $v \in Q_i$ for some $i \in \{1, 2\}$, and $(u, v) \notin E_r$, there exists a canonical graph $G(P \cup \{t\}, Q \cup \{s\}, E) \in \mathcal{G}(d)$ such that that $E_r \cup (u, v) \subseteq E$.

Proof: Let $G'(P \cup \{t\}, Q \cup \{s\}, E')$ be the graph obtained by adding edge (u, v) to G_r, that is, $E' = E_r \cup \{(u, v)\}$. Since u and v are free vertices, all vertex degrees in G' remain bounded by d. We now show how G' can be extended to a d-regular canonical graph.

We first add to G' a perfect matching M' of size d connecting an arbitrary set of d *free* vertices in Q_1 to an arbitrary set of d *free* vertices in P_2. This is feasible since G' has at least $\frac{5d}{2}$ free vertices each in both Q_1 and P_2. In the resulting graph, since the total degree of all vertices in P_i is same as the total degree of all vertices in Q_i, we can repeatedly pair together a vertex of degree less than d in P_i with a vertex of degree less than d in Q_i until degree of each vertex becomes exactly d, for $i \in \{1, 2\}$. Let E'' be the set of edges added to $G' \cup M'$ in this manner, and let G'' be the final graph. The graph G'' satisfies all properties of a canonical graph in the family $\mathcal{G}(d)$ except that it may not be a simple graph. We next transform G'' into a simple d-regular graph by suitably modifying edges in E''.

Given any graph $H(V_H, E_H)$, we define

$$\Phi(H) = \sum_{(x,y) \in V_H \times V_H} \max\{0, \eta(x,y) - 1\},$$

where $\eta(x,y)$ denotes the number of times the edge (x,y) appears in H. Note that $\Phi(H) = 0$ iff H is a simple graph. Consider any edge (a,b) that has multiplicity more than one in G''. It must be that $(a,b) \in E''$ since G' is a simple graph. Assume w.l.o.g. that $a \in P_1$ and $b \in Q_1$. Let $X \subset P_1$ and $Y \subset Q_1$ respectively denote the set of vertices adjacent to b and a in G''. Using condition (d) on the graph G_r, we know that

$$|E'' \cap (P_1 \times Q_1)| \geq \left(\frac{5d}{2}\right)\left(\frac{4d}{5} + 1\right) - (d+1) > 2d^2.$$

Since $|X| < d$ and $|Y| < d$, it follows that there must exist an edge $(a', b') \in E'' \cap (P_1 \times Q_1)$ such that $a' \notin X$ and $b' \notin Y$. We can thus replace edges $\{(a,b),(a',b')\}$ in E'' with edges $\{(a,b'),(a',b)\}$ without violating the d-regularity condition. It is easy to verify that the exchange reduces $\Phi(G'')$ by at least one, and that all edges involved in the exchange belong to the set E''. We can thus repeat this process until the graph G'' becomes simple, and hence a member of the family $\mathcal{G}(d)$. ■

Adversary strategy: For each vertex $u \in P \cup \{t\}, Q \cup \{s\}$, the adversary $\mathcal{A}$ maintains a list $N(u)$ of vertices adjacent to u that have been so far revealed to the algorithm $\mathcal{D}$. Wlog we can assume that the algorithm $\mathcal{D}$ never queries a vertex u for which $|N(u)| = d$. At any step of the game, we denote by G_r the graph formed by the edges revealed thus far. We say the game is in *evasive* mode if the graph G_r satisfies the condition (a) through (d) of Lemma 17, and is in *non-evasive* mode otherwise. Note that the game always starts in the evasive mode, and then switches to non-evasive mode.

When the game is in the evasive mode, in response to a query $\mathcal{Q}(u)$ by $\mathcal{D}$ for some free vertex $u \in P_i$ $(i \in \{1,2\})$, $\mathcal{A}$ returns an arbitrary free vertex $v \in Q_i$ such that $v \notin N(u)$. The adversary then adds v to $N(u)$ and u to $N(v)$. Similarly, when $\mathcal{D}$ asks a query $\mathcal{Q}(u)$ for some free vertex $u \in Q_i$ $(i \in \{1,2\})$, $\mathcal{A}$ returns an arbitrary free vertex $v \in P_i$ such that $v \notin N(u)$. It then adds v to $N(u)$ and u to $N(v)$ as above.

As the game transitions from evasive to non-evasive mode, Lemma 17 ensures existence of a canonical graph $G \in \mathcal{G}(d)$ that contains the graph revealed by the adversary thus far as a subgraph. The adversary answers all subsequent queries by $\mathcal{D}$ in a manner that is consistent with the edges of G. The lemma below shows that the simple adversary strategy above forces $\Omega(d^2)$ queries before the evasive mode terminates.

Lemma 18 *The algorithm makes $\Omega(d^2)$ queries before the game enters non-evasive mode.*

Proof: The adversary strategy ensures that conditions (a) through (c) in Lemma 17 are maintained at all times as long as the game is in the evasive mode. So we consider the first time that condition (d) is violated. Since each query answered by the adversary in the evasive mode contributes 1 to the

degree of exactly one vertex in $Q_1 \cup P_2$, $\mathcal{A}$ always answers at least $\Omega(d^2)$ queries before the number of vertices with degree less than $\frac{d}{5}$ falls below $\frac{5d}{2}$ in either Q_1 or P_2. The lemma follows. ■

Since A can not discover an edge in M' until the game enters the non-evasive mode, we obtain the desired lower bound of $\Omega(d^2)$.

2.9 An $\Omega(n \log n)$ High Probability Lower Bound For Dense Graphs

In this section we give a lower bound on the running time of any randomized algorithm for finding a perfect matching in d-regular bipartite multigraphs, even with edge multiplicities bounded above by $d/2$:

Theorem 19 *Let A be any randomized algorithm that finds a matching in a d-regular bipartite multigraph with n nodes and edge multiplicities bounded above by $d/2$. Then there exists a family of dense graphs for which A probes at least $(\gamma/64)n \ln n$ locations in the input adjacency arrays with probability at least $n^{-\gamma}$.*

We first reiterate that even though the algorithm obtained in section 2.6 is stated for simple graphs, the same runtime analysis applies for multigraphs as long as edge multiplicities are bounded above by $d/2$. The restriction on maximum edge multiplicity is necessary to ensure that SAMPLE-OUT-EDGE takes $O(1)$ time in expectation. In this section we show that every algorithm that finds a matching in a d-regular multigraph (even with edge multiplicities bounded above by $d/2$) probes at least $(\gamma/64)n \ln n$ locations in the input adjacency arrays with probability at least $n^{-\gamma}$ (on some fixed family of distributions). Thus, the lower and upper bounds are tight in this particular case. The lower bound instances use $d = \Theta(n)$.

It is also interesting to contrast Theorem 19 with the result of [92], which shows that sampling a constant number of edges incident to every vertex of a *complete* bipartite graph yields a subgraph that contains a perfect matching with high probability, i.e. the sampling complexity is $O(n)$ even if a high probability result is desired. The lower bound on the randomized algorithm is not as comprehensive as the deterministic lower bound: it holds only for very specific values of d (specifically, $d = \Theta(n)$), it bounds the "with high probability"-running time as opposed to the expected running time, and it works for multi-graphs. Obtaining tight upper and lower bounds for the entire range of parameters and for expected running time remains an interesting open problem.

We first introduce the following problem, which we will refer to as BIPARTITE-DISCOVERY(d).

Definition 20 *(BIPARTITE-DISCOVERY(d)) Let $G = (P, Q, E)$ be a bipartite multigraph with $|P| = 4d$ and $|Q| = d$. The set of edges $E(G)$ is constructed as follows. For each $u \in Q$ choose d neighbors in P uniformly at random with replacement. A node $u^* \in Q$ is then marked as special, and edges incident to the special node are referred to as special. The graph G is presented in adjacency array format with edges appearing in random order in adjacency lists.*

When an algorithm A queries a neighbor of a vertex $u \in Q$ or $v \in P$, the location of the edge in the adjacency arrays of both endpoints is revealed to A, i.e. it is no longer considered undiscovered when any of its endpoints is queried. The algorithm is not allowed to query the special node directly.

Algorithm A solves BIPARTITE-DISCOVERY(d) if it finds an edge incident to the special node. The cost of A is defined as the number of queries that it makes before discovering an edge to the special node.

We show the following:

Lemma 21 *Any algorithm that solves BIPARTITE-DISCOVERY(d) makes at least $(\gamma/2)d \ln d$ queries with probability at least $d^{-\gamma}$ for any $\gamma > 0$, where the probability is taken over the coin tosses used to generate the instance BIPARTITE-DISCOVERY.*

Proof: We first note that when an algorithm queries a neighbor of a vertex $u \in Q$ or $v \in P$, in fact an incident edge is returned uniformly at random among the yet undiscovered edges incident on u (resp. v), irrespective of the set of edges discovered by the algorithm so far.

Suppose that the algorithm has discovered J edges of G. Then the probability of the next query not yielding a special edge is at least $\frac{d^2-d-J}{d^2-J}$, independent of the actual set of edges of G that have already been discovered. Hence, the probability of not discovering a special edge after $J < d^2/3$ queries is at least

$$\prod_{j=0}^{J} \frac{d^2-d-j}{d^2-j} \geq \prod_{j=0}^{J} \frac{2d^2/3-d}{2d^2/3} \geq e^{-2J/d}$$

for sufficiently large d. Hence, we have that the probability of not finding a special edge after $(\gamma/2)d \ln d$ queries is at least $d^{-\gamma}$. We stress here that this probability is over the coin tosses used to generate the instance of BIPARTITE-DISCOVERY and not the coin tosses of the algorithm (which, in particular, could be deterministic). ■

We now give a reduction from BIPARTITE-DISCOVERY(d) to the problem of finding a matching in a regular bipartite multigraph with edge multiplicities bounded by $d/2$:

Proof of Theorem 19:

Let A be an algorithm that finds a matching in a regular bipartite multigraph with edge multiplicities bounded above by $d/2$ and makes fewer than $(\gamma/64)n \ln n$ queries with probability at least $1-n^{-\gamma}$ on every such graph. We will give an algorithm A' that solves BIPARTITE-DISCOVERY(d) and makes fewer than $(\gamma/2)d \ln d$ queries with probability strictly larger than $1 - d^{-\gamma}$.

Consider an instance $G = (P, Q, E)$ of BIPARTITE-DISCOVERY(d). Algorithm A' first checks if the degrees of all nodes in P are smaller than $d/2$. If there exists a node with degree strictly larger than $d/2$, A' queries all edges of all vertices in P and thus finds a special edge in at most $2d^2$ queries. Note that since the expected degree of vertices in P is $d/4$, the probability of this happening is at most e^{-d} for sufficiently large d by an application of the Chernoff bound with a union bound over vertices of P.

Now suppose that degrees of all nodes in P are at most $d/2$. A' adds a set of $3d$ vertices Q' to the Q side of the partition of G and connects nodes in Q' to nodes in P to ensure that the degree of every vertex in P and Q' is exactly d (it can be shown using an argument similar to the one in the proof of Lemma 17 that this can be done without introducing double edges). Denote the resulting regular multigraph by $G^+ = (P, Q \cup Q', E \cup E')$. Note that G^+ has $4d$ vertices in each part, and one vertex in the Q part of the bipartition is marked special together with its d adjacent edges. Now A' constructs the final graph by putting together two copies of G^+. In particular, we denote by G^- a mirrored copy of G^+, i.e. $G^- = (Q \cup Q', P, E \cup E')$, and finally denote by G^* the graph obtained by taking the union of G^+ and G^-, removing the two special nodes and identifying special edges in G^+ with special edges in G^-. Note that any matching in G^* contains a special edge, so algorithm A necessarily finds a special edge. Note that a query to an adjacency list in G^+ or G^- can be answered by doing at most one query on G. The number of vertices in each bipartition of G^* is $8d - 1$ and the degree of each node is d.

By assumption, algorithm A does not make more than $(\gamma'/64)n \ln n$ queries with probability at least $1 - n^{-\gamma'}$ for any $\gamma' > 0$. Setting $n = 8d - 1$ and $\gamma' = 2\gamma$, we get that A does not need more than $(2\gamma/64)8d \ln(8d) \leq (\gamma/2)d \ln d$ queries with probability at least $1 - (8d-1)^{-2\gamma} \geq 1 - d^{-2\gamma}$ for sufficiently large d. Hence, we conclude that A' probes at most $(\gamma/2)d \ln d$ locations with probability at least $1 - d^{-2\gamma} + e^{-d} > 1 - d^{-\gamma}$, contradicting Lemma 21. ■

Chapter 3

Matching covers and streaming

In this chapter we study the communication and streaming complexity of the maximum bipartite matching problem. Consider the following scenario. Alice holds a graph $G_A = (P, Q, E_A)$ and Bob holds a graph $G_B = (P, Q, E_B)$, where $|P| = |Q| = n$. Alice is allowed to send Bob a message m that depends only on the graph G_A. Bob must then output a matching $M \subseteq E_A \cup E_B$. What is the minimum size of the message m that Alice sends to Bob that allows Bob to recover a matching of size at least $1 - \epsilon$ of the maximum matching in $G_A \cup G_B$? The minimum message length is the *one-round communication complexity* of approximating bipartite matching, and is denoted by $CC(\epsilon, n)$. It is easy to see that the quantity $CC(\epsilon, n)$ also gives a lower bound on the space needed by a one-pass streaming algorithm to compute a $(1 - \epsilon)$-approximate bipartite matching. To see this, consider the graph $G_A \cup G_B$ revealed in a streaming manner with edge set E_A revealed first (in some arbitrary order), followed by the edge set E_B. It is clear that any non-trivial approximation to the bipartite matching problem requires $\Omega(n)$ communication and $\Omega(n)$ space, respectively, for the one–round communication and one-pass streaming problems described above. The central question considered in this work is how well can we approximate the bipartite matching problem when only $\tilde{O}(n)$ communication/space is allowed.

Matching Covers: We show that a study of these questions is intimately connected to existence of sparse "matching covers" for bipartite graphs. An *ϵ-matching cover* or simply an ϵ-cover, of a graph $G(P, Q, E)$ is a subgraph $G'(P, Q, E')$ such that for any pairs of sets $A \subseteq P$ and $B \subseteq Q$, the graph G' preserves the size of the largest A to B matching to within an additive error of ϵn. The notion of matching sparsifiers may be viewed as a natural analog of the notion of cut-preserving sparsifiers which have played a very important role in the study of network design and connectivity problems [52, 16]. It is easy to see that if there exists an ϵ-cover of size $f(\epsilon, n)$ for some function f, then Alice can just send a message of size $f(\epsilon, n)$ to allow Bob to compute an additive ϵn error approximation to bipartite matching (and $(1-\epsilon)$-approximation whenever $G_A \cup G_B$ contains a perfect matching). However, we show that the question of constructing efficient ϵ-covers is essentially equivalent to resolving a long-standing problem on a family of graphs known as the *Ruzsa-Szemerédi graphs*. A bipartite graph $G(P, Q, E)$ is an *ϵ-Ruzsa-Szemerédi graph* if E can be partitioned into a collection of induced matchings of size at least ϵn each. Ruzsa-Szemerédi graphs have been extensively studied as they arise naturally in property testing, PCP constructions and additive combinatorics [29, 43, 87]. A major open problem is to determine the maximum number of edges possible in an ϵ-Ruzsa-Szemerédi graph. In particular, do there exist dense graphs with large locally sparse regions (i.e. large induced subgraphs are perfect matchings)? We establish the following somewhat surprising relationship between matching covers and Ruzsa-Szemerédi graphs: for any $\epsilon > 0$ the smallest possible size of an ϵ-matching cover is essentially equal to the largest possible number of edges in an ϵ-Ruzsa-Szemerédi graph.

Constructing dense ϵ-Ruzsa-Szemerédi graphs for general ϵ and proving upper bounds on their size appears to be a difficult problem [40]. To our knowledge, there are two known constructions in the literature. The original construction due to Ruzsa and Szemerédi yields a collection of $n/3$ induced matchings of size $n/2^{O(\sqrt{\log n})}$ using Behrend's construction of a large subset of $\{1, \ldots, n\}$ without three-term arithmetic progressions [14, 87]. Constructions of a collection of $n^{c/\log\log n}$

induced matchings of size $n/3 - o(n)$ were given in [29, 77]. We use the ideas of [29, 77] to construct $(\frac{1}{2} - \delta)$-Ruzsa-Szemerédi graphs with $n^{1+\Omega_\delta(1/\log\log n)}$ edges and a more general construction for the vertex arrival case. To the best of our knowledge, the only known upper bound on the size of ϵ-Ruzsa-Szemerédi graphs for constant $\epsilon < \frac{1}{2}$ is $O(n^2/\log^* n)$ that follows from the bound used in an elementary proof of Roth's theorem [87].

One-round Communication: We show that in fact $CC(\epsilon, n) \leq 2n-1$ for all $\epsilon \geq \frac{1}{3}$, i.e. a message of linear size suffices to get a $\frac{2}{3}$-approximation to the maximum matching in $G_A \cup G_B$. We establish this result by constructing an $O(n)$ size $\frac{1}{2}$-cover of the input graph that satisfies certain additional properties which allows Bob to recover a $\frac{2}{3}$-approximation[1]. We refer to this particular $\frac{1}{2}$-cover as a *matching skelton* of the input graph, and give a polynomial time algorithm for constructing it. Next, building on the above-mentioned connection between matching covers and Ruzsa-Szemerédi graphs, we show the following two results: (a) our construction of $\frac{1}{2}$-cover implies that for any $\delta > 0$, there do not exist $(\frac{1}{2}+\delta)$-Ruzsa-Szemerédi graph with more than $O(n/\delta)$ edges, and (b) our $\frac{2}{3}$-approximation result is best possible when only linear amount of communication is allowed. In particular, Alice needs to send $n^{1+\Omega(1/\log\log n)}$ bits to achieve a $(\frac{2}{3}+\delta)$-approximation, for any constant $\delta > 0$, even when randomization is allowed.

We then study the one round communication complexity $CC_v(\epsilon, n)$ of $(1-\epsilon)$-approximate maximum matching in the restricted model when the graphs G_A and G_B are only allowed to share vertices on one side of the bipartition. This model is motivated by application to one-pass streaming computations when the vertices of the graph arrive together with all incident edges. We obtain a stronger approximation result in this model, namely, using the preceding $\frac{1}{2}$-cover construction we show that $CC_v(\epsilon, n) \leq 2n - 1$ for $\epsilon \geq 1/4$. Thus a $\frac{3}{4}$-approximation can be obtained with linear communication complexity, and as before, we show that obtaining a better approximation requires a communication complexity of $n^{1+\Omega(1/\log\log n)}$ bits.

One-pass Streaming: We build on our techniques for one-round communication to design a one-pass streaming algorithm for the case when vertices on one side are known in advance, and the vertices on the other side arrive in a streaming manner together with all their incident edges. This is precisely the setting of the celebrated $(1 - \frac{1}{e})$-competitive randomized algorithm of Karp-Vazirani-Vazirani (KVV) for the *online* bipartite matching problem [56]. We give a *deterministic* one-pass streaming algorithm that matches the $(1 - \frac{1}{e})$-approximation guarantee of KVV using only $O(n)$ space. Prior to our work, the only known *deterministic* algorithm for matching in one-pass streaming model, even under the assumption that vertices arrive together with all their edges, is the trivial algorithm that keeps a maximal matching, achieving a factor of $\frac{1}{2}$. We note that in the online setting, randomization is crucial as no deterministic online algorithm can achieve a competitive ratio better than $\frac{1}{2}$.

Related work: The streaming complexity of maximum bipartite matching has received significant attention recently. Space-efficient algorithms for approximating maximum matchings to factor $(1 - \epsilon)$ in a number of passes that only depends on $1/\epsilon$ have been developed. The work of [68]

[1] We note here that a maximum matching in a graph is only a $\frac{2}{3}$-cover.

gave the first space-efficient algorithm for finding matchings in general (non-bipartite) graphs that required a number of passes dependent only on $1/\epsilon$, although the dependence was exponential. This dependence was improved to polynomial in [26], where $(1-\epsilon)$-approximation was obtain in $O(1/\epsilon^8)$ passes. In a recent work, [3] obtained a significant improvement, achieving $(1-\epsilon)$-approximation in $O(\log\log(1/\epsilon)/\epsilon^2)$ passes (their techniques also yield improvements for the weighted version of the problem). Further improvements for the non-bipartite version of the problem have been obtained in [4]. Despite the large body of work on the problem, the only known algorithm for one pass is the trivial algorithm that keeps a maximal matching. No non-trivial lower bounds on the space complexity of obtaining constant factor approximation to maximum bipartite matching in one pass were known prior to our work (for exact computation, an $\Omega(n^2)$ lower bound was shown in [28]).

Organization: We start by introducing relevant definitions in section 3.1. In section 3.2 we give the construction of the *matching skeleton*, which we use later in section 3.3 to prove that $CC(1/3, n) = O(n)$, as well as show that the matching skeleton forms a 1/2-cover. In section 3.4 we deduce using the matching skeleton that $CC_v(1/4, n) = O(n)$. In section 3.5 we use these techniques to obtain a deterministic one-pass $(1 - 1/e)$ approximation to maximum matching in $O(n)$ space in the vertex arrival model. We extend the construction of Ruzsa-Szemerédi graphs from [29, 77] in section 3.6. We use these extensions in section 3.7 to show that our upper bounds on $CC(\epsilon, n)$ and $CC_v(\epsilon, n)$ are best possible, as well as to prove lower bounds on the space complexity of one-pass algorithms for approximating maximum bipartite matching. Finally, in section 3.8 we prove the correspondence between the size of the smallest ϵ-matching cover of a graph on n nodes and the size of the largest ϵ-Ruzsa-Szemerédi graph on n nodes.

3.1 Preliminaries

We start by defining bipartite matching covers, which are matchings-preserving graph sparsifiers.

Definition 22 *Given an undirected bipartite graph $G = (P, Q, E)$, and sets $A \subseteq P, B \subseteq Q$, and $H \subseteq E$, let $M_H(A, B)$ denote the size of the largest matching in the graph $G' = (A, B, (A \times B) \cap H)$.*

Given an undirected bipartite graph $G = (P, Q, E)$ with $|P| = |Q| = n$, a set of edges $H \subseteq E$ is said to be an *ϵ-matching-cover* of G if for all $A \subseteq P, B \subseteq Q$, we have $M_H(A, B) \geq M_E(A, B) - \epsilon n$.

Definition 23 *Define $L_C(\epsilon, n)$ to be the smallest number m' such that any undirected bipartite graph $G = (P, Q, E)$ with $P = Q = n$ has an ϵ-matching-cover of size at most m'.*

We next define induced matchings and Ruzsa-Szemerédi graphs.

Definition 24 *Given an undirected bipartite graph $G = (P, Q, E)$ and a set of edges $F \subseteq E$, let $P(F) \subseteq P$ denote the set of vertices in P which are incident on at least one edge in F, and analogously, let $Q(F)$ denote the set of vertices in Q which are incident on at least one edge in F. Let $E(F)$, called the set of edges induced by F, denote the set of edges $E \cap (P(F) \times Q(F))$. Note that $E(F)$ may be much larger than F in general.*

Given an undirected bipartite graph $G = (P, Q, E)$, a set of edges $F \subseteq E$ is said to be an *induced matching* if no two edges in F share an endpoint, and $E(F) = F$. Given an undirected bipartite graph $G = (P, Q, E)$ and a partition $\mathcal{F}$ of E, the partition is said to be an *induced partition* of G if every set $F \in \mathcal{F}$ is an induced matching. An undirected bipartite graph $G = (P, Q, E)$ with $P = Q = n$ is said to have an *ϵ-induced partition* if there exists an induced partition of G such every set in the partition is of size at least ϵn. Following [29], we refer to graphs that have an ϵ-induced partition as *ϵ-Ruzsa-Szemerédi graphs*.

Definition 25 *Let $U_I(\epsilon, n)$ denote the largest number m such that there exists an undirected bipartite graph $G = (P, Q, E)$ with $|E| = m, |P| = |Q| = n$, and with an ϵ-induced partition.*

Note that for any $0 < \epsilon_1 < \epsilon_2 < 1$, any ϵ_2-induced partition of a graph is also an ϵ_1-induced partition, and hence, $U_I(\epsilon, n)$ is a non-increasing function of ϵ. Analogously, any ϵ_1-matching-cover is also an ϵ_2-matching cover, and hence, $L_C(\epsilon, n)$ is also a non-increasing function of ϵ.

3.2 Matching Skeletons

Let $G = (P, Q, E)$ be a bipartite graph. We now define a subgraph $G' = (P, Q, E')$ of G that contains at most $(|P| + |Q| - 1)$ edges, and encodes useful information about matchings in G. We refer to this subgraph G' as a *matching skeleton* of G, and this construction will serve as a building block for our algorithms. Among other things, we will show later that G' is a $\frac{1}{2}$-cover of G.

We present the construction of G' in two steps. We first consider the case when P is *hypermatchable*, that is, for every vertex $v \in Q$ there exists a perfect matching of the P side that does not include v. We then extend the construction to the general case using the Edmonds-Gallai decomposition [81].

3.2.1 P is hypermatchable in G

We note that since P is *hypermatchable*, by Hall's theorem [81], we have that $|\Gamma(A)| > |A|$ for all $A \subseteq P$. For a parameter $\alpha \in (0, 1]$, let $\mathcal{R}_G(\alpha) = \{A \subseteq P : |\Gamma_G(A)| \leq (1/\alpha)|A|\}$. Note that as the parameter α *decreases*, the expansion requirement in the definition above *increases*. We will omit the subscript G when G is fixed, as in the next lemma.

Lemma 26 *Let $\alpha \in (0, 1]$ be such that $\mathcal{R}(\alpha+\epsilon) = \emptyset$ for any $\epsilon > 0$, i.e. G supports an $\frac{1}{\alpha+\epsilon}$-matching of the P-side for any $\epsilon > 0$. Then for any two $A_1 \in \mathcal{R}(\alpha), A_2 \in \mathcal{R}(\alpha)$ one has $A_1 \cup A_2 \in \mathcal{R}(\alpha)$.*

Proof: Let $B_1 = \Gamma(A_1)$ and $B_2 = \Gamma(A_2)$. First, since $(A_1 \times (Q \setminus B_1)) \cap E = \emptyset$ and $(A_2 \times (Q \setminus B_2)) \cap E = \emptyset$, we have that $(A_1 \cap A_2) \times (Q \setminus (B_1 \cap B_2)) = \emptyset$. Furthermore, since $\mathcal{R}(\alpha + \epsilon) = \emptyset$, one has $|B_1 \cap B_2| \geq (1/\alpha)|A_1 \cap A_2|$. Also, we have $|B_i| \leq |A_i|/\alpha, i = 1, 2$. Hence,

$$|B_1 \cup B_2| = |B_1| + |B_2| - |B_1 \cap B_2| \leq (1/\alpha)(|A_1| + |A_2| - |A_1 \cap A_2|) = (1/\alpha)|A_1 \cup A_2|,$$

and thus $(A_1 \cup A_2) \in \mathcal{R}(\alpha)$ as required. ■

We now define a collection of sets $(S_j, T_j), j = 1, \ldots, +\infty$, where $S_j \subseteq P, T_j \subseteq Q, S_i \cap S_j = \emptyset, i \neq j$.

1. Set $j := 1$, $G_0 := G, \alpha_0 := 1$. We have $\mathcal{R}_{G_0}(\alpha_0) = \emptyset$.
2. Let $\beta < \alpha_{j-1}$ be the largest real such that $\mathcal{R}_{G_{j-1}}(\beta) \neq \emptyset$.
3. Let $S_\beta = \bigcup_{A \in \mathcal{R}(\beta)} A$, and $T_\beta = \Gamma(S_\beta)$. We have $S_\beta \in \mathcal{R}_{R_{j-1}}(\beta)$ by Lemma 26.
4. Let $G_j := G_{j-1} \setminus (S_\beta \cup T_\beta)$. We refer to the value of α at which a pair (S_α, T_α) gets removed from the graph as the expansion of the pair. Set $S_j := S_\beta, T_j := T_\beta, \alpha_j := \beta$. If $G_j \neq \emptyset$, let $j := j + 1$ and go to (2).

The following lemma is an easy consequence of the above construction.

Lemma 27 *1. For each $U \subseteq S_j$ one has $|\Gamma_{G_j}(U)| \geq (1/\alpha_j)|U|$.*

2. For every $k > 0$, $\left(\left(\bigcup_{j \leq k} S_j\right) \times \left(Q \setminus \bigcup_{j \leq k} T_j\right)\right) \cap E = \emptyset$.

Proof: We prove (1) by contradiction. When $j = 1$, (1) follows immediately since we are choosing the largest β such that $\mathcal{R}(\beta) \neq \emptyset$. Otherwise suppose that there exists $U \subseteq P_{G_j}$ such that $|\Gamma_{G_j}(U)| < (1/\alpha_j)|U|$. Then first observe that $|\Gamma_{G_j}(U)| > (1/\alpha_{j-1})|U|$. If not then

$$|\Gamma_{G_{j-1}}(S_{j-1} \cup U)| = |T_{j-1}| + |\Gamma_{G_j}(U)| \leq \frac{1}{\alpha_{j-1}}(|S_{j-1}| + |U|) \leq \frac{1}{\alpha_{j-1}}(|S_{j-1} \cup U|),$$

since $S_{j-1} \cap P_{G_j} = \emptyset$ by construction. Now as $\alpha_j < \alpha_{j-1}$ is chosen to be the largest real for which there exists some subset $U' \subseteq P_{G_j}$ with $|\Gamma_{G_j}(U')| \leq (1/\alpha_j)|U'|$, it follows that for every $U \subseteq P_{G_j}$, we must have $|\Gamma_{G_j}(U)| \geq (1/\alpha_j)|U|$.

(2) follows by construction.

■

To complete the definition of the matching skeleton, we now identify the set of edges of G that our algorithm keeps. For a parameter $\gamma \geq 1$ and subsets $S \subseteq P$, $T \subseteq Q$ we refer to a (fractional) matching M that saturates each vertex in S exactly γ times (fractionally) and each vertex in T at most once as a *γ-matching* of S in $(S, T, (S \times T) \cap E)$. By Lemma 27 there exists a (fractional) $(1/\alpha_j)$-matching of S_j in $(S_j, T_j, (S_j \times T_j) \cap E)$. Moreover, one can ensure that the matching is supported on the edges of a forest by rerouting flow along cycles. Let M_j be a fractional $(1/\alpha_j)$-matching in (S_j, T_j) that is a forest.

Interestingly, the fractional matching corresponding to the matching skeleton is identical to a 1-majorized fractional allocation of unit-sized jobs to $(1 - \infty)$ machines [59, 37]; as a result, the fractional matchings x_e simultaneously minimize all convex functions of the x_e's subject to the constraint that every node in P is matched exactly once.

3.2.2 General bipartite graphs

We now extend the construction to general bipartite graphs using the Edmonds-Gallai decomposition of $G(P, Q, E)$, which essentially allows us to partition the vertices of G into sets $A_P(G)$, $D_P(G)$, $C_P(G)$, $A_Q(G)$, $D_Q(G)$, and $C_Q(G)$ such that $A_P(G)$ is hypermatchable to $D_Q(G)$, A_Q is hypermatchable to $D_P(G)$, and there is a perfect matching between $C_P(G)$ and $C_Q(G)$.

The Edmonds-Gallai decomposition theorem is as follows.

Theorem 28 *(Edmonds-Gallai decomposition, [81]) Let $G = (V, E)$ be a graph. Then V can be partitioned into the union of sets $D(G), A(G), C(G)$ such that*

$$\begin{aligned} D(G) &= \{v \in V | \textit{there exists a maximum matching missing } v\} \\ A(G) &= \Gamma(D(G)) \\ C(G) &= V \setminus (D(G) \cup A(G)). \end{aligned}$$

Moreover, every maximum matching contains a perfect matching inside $C(G)$.

Applying Edmonds-Gallai decomposition to bipartite graphs, we get

Corollary 29 *Let $G = (P, Q, E)$ be a graph. Then V can be partitioned into the union of sets $D_P(G), D_Q(G)$, $A_P(G), A_Q(G)$, $C_P(G), C_Q(G)$ such that*

$$\begin{aligned} D_P(G) &= \{v \in P | \textit{there exists a maximum matching missing } v\} \\ D_Q(G) &= \{v \in Q | \textit{there exists a maximum matching missing } v\} \\ A_P(G) &= \Gamma(D_Q(G)) \\ A_Q(G) &= \Gamma(D_P(G)) \\ C_P(G) &= P \setminus (D_P(G) \cup A_P(G)) \\ C_Q(G) &= Q \setminus (D_Q(G) \cup A_Q(G)). \end{aligned}$$

Moreover,

1. *there exists a perfect matching between $C_P(G)$ and $C_Q(G)$*
2. *for every $U \subseteq A_P(G)$ one has $|\Gamma(U) \cap D_Q(G)| > |U|$*
3. *for every $U \subseteq A_Q(G)$ one has $|\Gamma(U) \cap D_P(G)| > |U|$.*

Proof: (1) is part of the statement of Theorem 28. To show (2), note that by definition of $D_Q(G)$ for each vertex $v \in D_Q(G)$ there exists a maximum matching that misses v. Thus, $|\Gamma(U) \cap D_Q(G)| > |U|$ for every set U. ■

Using the above partition, we can now define a matching skeleton of G using the above partition. Let $S_0 = C_P(G), T_0 = C_Q(G)$, and let M_0 be a perfect matching between S_0 and T_0. Let $(S_1, T_1), \ldots,$ (S_j, T_j) be the expanding pairs obtained by the construction in the previous section on the graph induced by $A_P(G) \cup D_Q(G)$. Let $(S_{-j}, T_{-j}), \ldots, (S_{-1}, T_{-1})$ be the expanding pairs obtained by the construction in the previous section from the Q side on the graph induced by $A_Q(G) \cup D_P(G)$.

Definition 30 *For a bipartite graph $G = (P, Q, E)$ we define the matching skeleton G' of G as the union of pairs $(S_j, T_j), j = -\infty, \ldots, +\infty$, with corresponding (fractional) matchings M_j. Note that G' contains at most $|P| + |Q| - 1$ edges.*

As before, we can show the following:

Lemma 31 *1. For each $U \subseteq S_j$, one has $|T_j \cap \Gamma_{G'}(U)| \geq (1/\alpha_j)|U|$.*

2. For every $k > 0$, $\left(\left(P \setminus \bigcup_{j \geq k} S_j\right) \times \left(\bigcup_{j \geq k} T_j\right)\right) \cap E = \emptyset$, and $\left(\left(Q \setminus \bigcup_{j \leq -k} S_j\right) \times \left(\bigcup_{j \leq -k} T_j\right)\right) \cap E = \emptyset$.

Proof: Follows by construction of G'. ■

We note that the formulation of property (2) in Lemma 31 is slightly different from property (2) in Lemma 27. However, one can see that these formulations are equivalent when there are no (S_j, T_j) pairs for negative j, as is the case in Lemma 27.

3.3 $O(n)$ communication protocol for $CC(\frac{1}{3}, n)$

In this section, we prove that for any two bipartite graphs G_1, G_2, the maximum matching in the graph $G_1' \cup G_2$ is at least 2/3 of the maximum matching in $G_1 \cup G_2$, where G_1' is the matching skeleton of G_1. Thus, $CC(\epsilon, n) = O(n)$ for all $\epsilon \geq 1/3$; Alice sends the matching skeleton G_A' of her graph, and Bob computes a maximum matching in the graph $G_A' \cup G_B$.

Before proceeding, we establish some notation used for the next several sections. Denote by $(S_j, T_j), j = -\infty, \ldots, +\infty$ the set of pairs from the definition of G'. Recall that $S_j \subseteq P$ when $j \geq 0$ and $S_j \subseteq Q$ when $j < 0$. Also, given a maximum matching M in a bipartite graph $G = (P, Q, E)$, a *saturating cut* corresponding to M is a pair of disjoint sets $(A_1 \cup B_1, A_2 \cup B_2)$ such that $A_1 \cup A_2 = P, B_1 \cup B_2 = Q$, all vertices in $A_2 \cup B_1$ are matched by M, there are no matching edges between A_2 and B_1, and no edges at all between A_1 and B_2. The existence of a saturating cut follows from the max-flow min-cut theorem. Let ALG denote the size of the maximum matching in $G_1' \cup G_2$ and let OPT denote the size of the maximum matching in $G_1 \cup G_2$.

Consider a maximum matching M in $(G_1' \cup G_2)$ and a corresponding saturating cut $(A_1 \cup B_1, A_2 \cup B_2)$; note that $ALG = |B_1| + |A_2|$. Let M^* be a maximum matching in $E_1 \cap (A_1 \times B_2)$. Note that we have $OPT \leq |B_1| + |A_2| + |M^*|$.

We start by describing the intuition behind the proof. Suppose for simplicity that the matching skeleton G_1' of G_1 consists of only one (S_j, T_j) pair for some $j \geq 0$, such that $|T_j| = (1/\alpha_j)|S_j|$. We first note that since the matching M^* is not part of the matching skeleton, it must be that edges of M^* go from S_j to T_j. We will abuse notation slightly by writing $M^* \cap X$ to denote, for $X \subseteq P \cup Q$, the subset of nodes of X that are matched by M^*. Since all edges of M^* go from S_j to T_j, we have $M^* \cap A_1 \subseteq S_j \cap A_1$ and $M^* \cap B_2 \subseteq T_j \cap B_2$. This allows us to obtain a lower bound on $|B_1|$ and $|A_2|$ in terms of $|M^*|$ if we lower bound $|B_1|$ and $|A_2|$ in terms of $|S_j \cap A_1|$ and $|T_j \cap B_2|$ respectively. First, we have that $|B_1| \geq |\Gamma_{G_1'}(S_j \cap A_1)| \geq (1/\alpha_j)|S_j \cap A_1| \geq (1/\alpha_j)|M^*|$, where we used the fact that the saturating cut is empty in $G_1' \cup G_2$ and Lemma 31 . Next, we prove that

$|\Gamma_{G'_1}(S_j \cap A_2) \cap B_2| \leq (1/\alpha_j)|S_j \cap A_2|$ (this is proved in Lemma 33 below). This, together with the fact that $M^* \cap B_2 \subseteq T_j \cap B_2 = \Gamma_{G'_1}(S_j \cap A_2) \cap B_2$, implies that $|A_2| \geq \alpha_j |M^*|$. Thus, we always have $|A_2| + |B_1| \geq (\alpha_j + 1/\alpha_j)|M^*|$, and hence the worst case happens at $\alpha_j = 1$, i.e. when the matching skeleton G'_1 of G_1 consists of only the (S_0, T_0) pair, yielding a 2/3 approximation. The proof sketch that we just gave applies when the matching skeleton only contains one pair (S_j, T_j). In the general case, we use Lemma 31 to control the distribution of M^* among different (S_j, T_j) pairs. More precisely, we use the fact that edges of M^* may go from $S_j \cap A_1$ to $T_i \cap B_2$ *only if* $i \leq j$. Another aspect that adds complications to the formal proof is the presence of (S_j, T_j) pairs for negative j.

We will use the notation

$$Z_j \subseteq \begin{cases} S_j \cap A_1, & j > 0 \\ S_j \cap B_2, & j < 0. \end{cases} \qquad \text{and} \qquad W_j \subseteq \begin{cases} T_j \cap B_2, & j > 0 \\ T_j \cap A_1, & j < 0 \end{cases}$$

for the vertices in P and Q that are matched by M^* (see Fig. 3.3). Further, let Z^* denote the set of vertices in $S_0 \cap A_1$ that are matched by M^* to $B_2 \cap T_0$, and let $W^* = M^*(Z^*) \subseteq B_2 \cap T_0$. Let $W_0^1 \subseteq S_0 \cap A_1$ denote the vertices in $S_0 \cap A_1$ that are matched by M^* outside of T_0. Similarly, let $W_0^2 \subseteq T_0 \cap B_2$ denote the vertices in $T_0 \cap B_2$ that are matched by M^* outside of S_0 (see Fig. 3.3). Let

$$B'_1 := B_1 \cap \left(\Gamma_{G'_1}(Z^*) \cup \Gamma_{G'_1}(W_0^1) \cup \bigcup_{j>0} \left(\Gamma_{G'_1}(Z_j) \cup S_{-j} \right) \right)$$

$$A'_2 := A_2 \cap \left(\Gamma_{G'_1}(W^*) \cup \Gamma_{G'_1}(W_0^2) \cup \bigcup_{j<0} \left(\Gamma_{G'_1}(Z_j) \cup S_{-j} \right) \right).$$

Then since

$$OPT \leq |B'_1| + |A'_2| + |M^*| + (|B_1 \setminus B'_1| + |A_2 \setminus A'_2|)$$
$$ALG = |B'_1| + |A'_2| + (|B_1 \setminus B'_1| + |A_2 \setminus A'_2|),$$

it is sufficient to prove that $(|B'_1| + |A'_2|) \geq (2/3)(|B'_1| + |A'_2| + |M^*|)$. Let $OPT' = |B'_1| + |A'_2| + |M^*|$ and $ALG' = |B'_1| + |A'_2|$. Define $\Delta' = (OPT' - ALG')/OPT'$.

We will now define variables to represent the sizes of the sets used in defining B'_1, A'_2:

$$w_0^1 = |W_0^1|, w_0^2 = |W_0^2|, z^* = |Z^*|, w^* = |W^*|, (\text{Note that } z^* = w^*)$$
$$z_j = |Z_j|, w_j = |W_j|, r_j = |\Gamma_{G'_1}(Z_j)|, s_j = \begin{cases} |S_j \cap A_2| & j > 0 \\ |S_j \cap B_1| & j < 0 \end{cases}.$$

Lemma 32 expresses the size of B'_1 and A'_2 in terms of the new variables defined above.

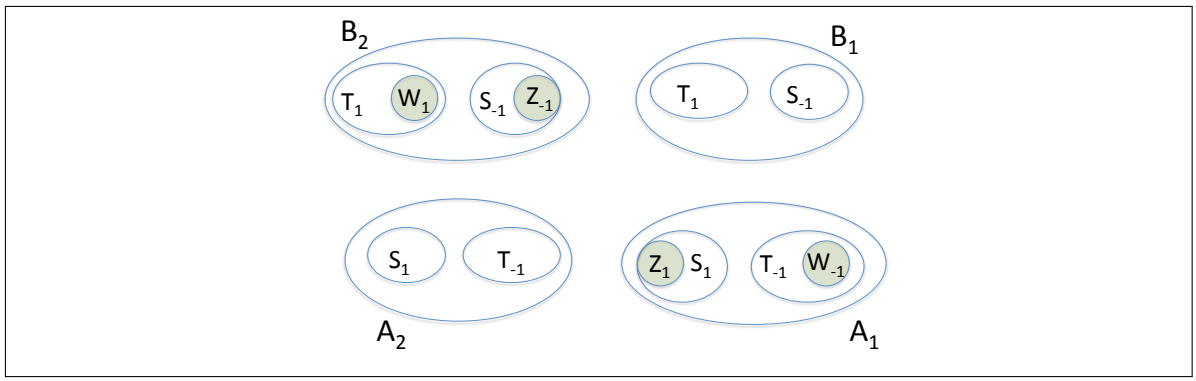


Figure 3.1: The structure of the saturating cut (sets Z^*, W^*, W_0^1, W_0^2)

.

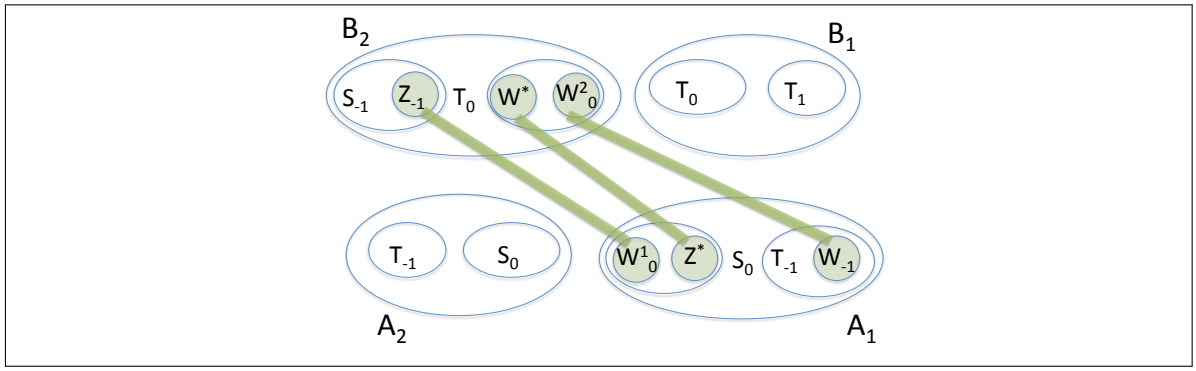


Figure 3.2: The structure of the saturating cut

.

Lemma 32 *$ALG' = \sum_{j\neq 0}(s_j + r_j) + (z^* + w_0^1) + (w^* + w_0^2)$, and $OPT' \leq z^* + (z^* + w_0^1) + (w^* + w_0^2) + \sum_{j\neq 0}(s_j + z_j + r_j)$.*

. **Proof:** The main idea is that most of the sets in the definitions of B_1' and A_2' are disjoint, allowing us to represent sizes of unions of these sets by sums of sizes of individual sets.

For ALG', recall that $\Gamma_{G_1'}(S_j) = T_j$ and hence, the sets $\Gamma_{G_1'}(S_j)$ are all disjoint. Further, the sets S_j are all disjoint, by construction, and disjoint with all the T_j's. Thus, $|A_1'| + |B_2'| = |\Gamma_{G_1'}(W^*) \cup \Gamma_{G_1'}(W_0^2)| + |\Gamma_{G_1'}(Z^*) \cup \Gamma_{G_1'}(W_0^1)| + \sum_{j\neq 0}(s_j + r_j)$. The sets W^* and W_0^2 are disjoint. Further, they are subsets of T_0 (corresponding to $\alpha = 1$), and hence nodes in these sets have a single unique neighbor in G_1'; consequently $|\Gamma_{G_1'}(W^*) \cup \Gamma_{G_1'}(W_0^2)| = w^* + w_0^2$. Similarly, $|\Gamma_{G_1'}(Z^*) \cup \Gamma_{G_1'}(W_0^1)| = z^* + w_0^1$. This completes the proof of the lemma for ALG'.

We have $OPT' = ALG' + |M^*|$. Consider any edge $(u, v) \in M^*$. This edge is not in G_1' and hence must go from an S_j to a $T_{j'}$ where $0 \leq j' \leq j$ or $0 \geq j' \geq j$. The number of edges in M^* that go from S_0 to T_0 is precisely z^* by definition; the number of remaining edges is precisely $\sum_{j\neq 0} z_j$. ■

We now derive linear constraints on the size variables, leading to a simple linear program. We have by Lemma 31 that for all $k > 0$

$$\left(\left(P \setminus \bigcup_{j\geq k} Z_j\right) \times \left(\bigcup_{j\geq k} W_j\right)\right) \cap E_1 = \emptyset, \quad \text{and} \quad \left(\left(Q \setminus \bigcup_{j\leq -k} Z_j\right) \times \left(\bigcup_{j\leq -k} W_j\right)\right) \cap E_1 = \emptyset. \tag{3.1}$$

The existence of M^* together with (3.1)yields

$$\sum_{j=k}^{+\infty} z_j \geq \sum_{j=k}^{+\infty} w_j, \forall k > 0, \qquad \text{and} \qquad \sum_{j=-\infty}^{-k} z_j \geq \sum_{j=-\infty}^{-k} w_j, \forall k > 0. \tag{3.2}$$

Furthermore, we have by definition of W_0^1 together with (3.1)that

$$w_0^1 \leq \sum_{j<0} z_j - \sum_{j<0} w_j \qquad \text{and} \qquad w_0^2 \leq \sum_{j>0} z_j - \sum_{j>0} w_j. \tag{3.3}$$

Also, we have

$$\sum_{j<0} z_j = w_0^1 + \sum_{j<0} w_j \qquad \text{and} \qquad \sum_{j>0} z_j = w_0^2 + \sum_{j>0} w_j. \tag{3.4}$$

Next, by Lemma 31, we have $r_j \geq (1/\alpha_j)z_j$. We also need

Lemma 33 *(1)* $|\Gamma_{G_1'}(S_j \cap A_2) \cap B_2| \leq (1/\alpha_j)|S_j \cap A_2|$ *for all* $j > 0$*, and (2)* $|\Gamma_{G_1'}(S_j \cap B_1) \cap A_1| \leq (1/\alpha_j)|S_j \cap B_1|$ *for all* $j < 0$*.*

Proof: We prove (1). The proof of (2) is analogous. Suppose that $|\Gamma_{G_1'}(S_j \cap A_2) \cap B_2| > (1/\alpha_j)|S_j \cap A_2|$. Then using the assumption that $(A_1 \times B_2) \cap E' = \emptyset$, we get

$$\begin{aligned} |T_j| &= |T_j \cap B_2| + |T_j \cap B_1| \geq |\Gamma_{G_1'}(S_j \cap A_2) \cap B_2| + |\Gamma_{G_1'}(S_j \cap A_1)| \\ &> (1/\alpha_j)|S_j \cap A_2| + (1/\alpha_j)|S_j \cap A_1| > (1/\alpha_j)|S_j|, \end{aligned}$$

a contradiction to the definition of the matching skeleton. ■

We will now bound $\Delta' = (OPT' - ALG')/OPT'$ using a sequence of linear programs, described in figures 3.3-3.4(b). We will overload notation to use P_1^*, P_2^*, P_3^*, respectively, to refer to these linear programs as well as their optimum objective function value. By Lemma 33 one has for all $j \neq 0$ that $(1/\alpha_j)s_j \geq w_j$. We combine this with equations 3.2, 3.3, and 3.4 to obtain the first of our linear programs, P_1^*, in figure 3.3. Bounding Δ' is equivalent to bounding this LP (i.e. $\Delta' \leq P_1^*$). Note that we have implicitly rescaled the variables so that $OPT' \leq 1$.

We now symmetrize the LP P_1^* by collecting the variables for cases when j is positive, negative, and 0 to obtain LP P_2^* in figure 3.4(a). Finally, we relax LP P_2^* by combining the second and third constraints, and then establish that the remaining constraints are all tight. This gives us the LP P_3^* in figure 3.4(b). Details of the construction are embedded in the proof of the following lemma.

$$P_1^* = \text{maximize } z^* + \sum_{j\neq 0} z_j \quad \text{s.t.}$$

$$z^* + (z^* + w_0^1) + (w^* + w_0^2) + \sum_{j\neq 0} s_j + z_j + r_j \leq 1$$

$$\forall k > 0, \sum_{j=k}^{+\infty} z_j \geq \sum_{j=k}^{+\infty} w_j,$$

$$\forall k > 0, \sum_{j=-\infty}^{-k} z_j \geq \sum_{j=-\infty}^{-k} w_j$$

$$\forall j \neq 0, (1/\alpha_j) s_j \geq w_j$$

$$\forall j \neq 0, r_j \geq (1/\alpha_j) z_j$$

$$\sum_{j<0} z_j = w_0^1 + \sum_{j<0} w_j$$

$$\sum_{j>0} z_j = w_0^2 + \sum_{j>0} w_j$$

$$z^* = w^*$$

$$s, z, w, r, z^*, w^*, w_0^1, w_0^2 \geq 0$$

Figure 3.3: The linear programs for lower bounding ALG/OPT (P_1^*).

$$P_2^* = \text{maximize } \sum_{j=0}^{+\infty} z_j \quad \text{s.t.}$$

$$\sum_{j=0}^{+\infty} s_j + z_j + r_j \leq 1$$

$$\forall k \geq 0, \sum_{j=0}^{k} w_j \geq \sum_{j=0}^{k} z_j$$

$$(1/\alpha_j) s_j \geq w_j, j \geq 0$$

$$r_j \geq (1/\alpha_j) z_j, j \geq 0$$

$$x, z, w, r \geq 0$$

$$P_3^* = \text{maximize } \sum_{j=0}^{\infty} z_j \quad \text{s.t.}$$

$$\sum_j (\alpha_j + 1 + 1/\alpha_j) z_j \leq 1$$

$$z \geq 0$$

Figure 3.4: The linear programs for lower bounding ALG/OPT (P_2^* and P_3^*).

Lemma 34 $P_1^* \leq P_2^* \leq P_3^*$.

Proof:

From P_1^* to P_2^*

We will show that the optimum of the LP P_2^* in figure 3.4(a) is an upper bound for the optimum of P_1^* in figure 3.3. First increase the set $\{\alpha_j\}_{j=-\infty}^{\infty}$ to ensure that $\alpha_j = \alpha_{-j}$(this can only improve the objective function). Now, we define

$$\begin{aligned} s'_j &= s_j + s_{-j}, j > 0 \\ r'_j &= r_j + r_{-j}, j > 0 \\ z'_j &= z_j + z_{-j}, j > 0 \\ w'_j &= w_j + w_{-j}, j > 0 \\ w'_0 &= w^* + w_0^1 + w_0^2 \\ s'_0 &= w^* + w_0^1 + w_0^2 \\ z'_0 &= z^* \\ r'_0 &= z^*. \end{aligned} \tag{3.5}$$

We will show that if $s, r, z, w, z^*, w^*, w_0^1, w_0^2$ are feasible for P_1^*, then s', r', z', w' are feasible for P_2^* with the same objective function value.

First, the objective function is exactly the same by inspection. Constraints 3 and 4 of P_2^* for $j > 0$ are linear in the respective variables and are hence satisfied. Furthermore, one has

$$(1/\alpha_0)s'_0 = w^* + w_0^1 + w_0^2 = w'_0$$

and

$$r'_0 = z^* = z'_0.$$

Hence, constraints 3 and 4 are satisfied for all $j \geq 0$.

To verify that constraint 1 is satisfied, we calculate

$$\begin{aligned} \sum_{j=0}^{+\infty} s'_j + z'_j + r'_j &= s'_0 + z'_0 + r'_0 + \sum_{j=1}^{+\infty}(s'_j + z'_j + r'_j) \\ &= (w^* + w_0^1 + w_0^2) + z^* + z^* + \sum_{j\neq 0}(s_j + z_j + r_j) \\ = z^* + (z^* + w_0^1) + (z^* + w_0^2) + \sum_{j\neq 0}(s_j + z_j + r_j) &\leq 1. \end{aligned}$$

We now verify that constraint 2 of P_2^* is satisfied. First, for $k = 0$ one has

$$w'_0 = w^* + w_0^1 + w_0^2 \geq w^* = z^* = z'_0.$$

Next, note that by adding constraints 2,3 of P_1^* we get

$$\sum_{|j|\geq k} z_j \geq \sum_{|j|\geq k} w_j \tag{3.6}$$

for all $k > 0$. Adding constraints 6 and 7 of P_1^*, we get

$$\sum_{j\neq 0} z_j = w_0^1 + w_0^2 + \sum_{j\neq 0} w_j. \tag{3.7}$$

Subtracting (3.7) from (3.6), we get

$$\sum_{|j|=1}^{k} z_j \leq w_0^1 + w_0^2 + \sum_{|j|=1}^{k} w_j. \tag{3.8}$$

Adding z^* to both sides and using the fact that $z_0' = z^*$ and $w_0' = z^* + w_0^1 + w_0^2$, we get

$$\sum_{j=0}^{k} z_j \leq \sum_{j=0}^{k} w_j. \tag{3.9}$$

This completes the proof of the first half of lemma 34.

From P_2^* to P_3^*

We now bound P_2^*. First we relax the constraints by adding constraint 3 of over j from 0 to k and adding to constraint 2:

$$\begin{aligned}
\text{maximize } & \sum_{j=0}^{\infty} z_j \\
\text{s.t.} & \\
& \sum_{j=0}^{\infty} s_j + z_j + r_j \leq 1 \\
& \sum_{j=0}^{k} (1/\alpha_j) s_j \geq \sum_{j=0}^{k} z_j, \forall k \geq 0 \\
& r_j \geq (1/\alpha_j) z_j, \forall j \geq 0 \\
& x, z, w, r \geq 0
\end{aligned} \tag{3.10}$$

Note that the first constraint is necessarily tight at the optimum. Otherwise scaling all variables to make the constraint tight increases the objective function. We now show that all of the constraints in the second line of (3.10) are necessarily tight at the optimum. Indeed, let $k^* \geq 0$ be the smallest

such that $\sum_{j=0}^{k^*}(1/\alpha_j)s_j > \sum_{j=0}^{k^*} z_j$. Note that one necessarily has $s_{k^*} > 0$. Let

$$s' = s - \delta e_{k^*} + (\alpha_{k^*+1}/\alpha_{k^*})\delta e_{k^*+1}$$
$$r' = r, z' = z,$$

where e_j denotes the vector of all zeros with 1 in position j. Then

$$\sum_{j=0}^{k}(1/\alpha_j)s'_j \geq \sum_{j=0}^{k} z'_j$$

for all k and

$$\sum_{j=0}^{\infty}(s'_j + z'_j + r'_j) = 1 - \delta(1 - \alpha_{k^*+1}/\alpha_{k^*}).$$

So for sufficiently small positive $\delta > 0$ one has that

$$s'' = s'/(1 - \delta(1 - \alpha_{k^*+1}/\alpha_{k^*}))$$
$$r'' = r'/(1 - \delta(1 - \alpha_{k^*+1}/\alpha_{k^*}))$$
$$z'' = z'/(1 - \delta(1 - \alpha_{k^*+1}/\alpha_{k^*}))$$

form a feasible solution with a better objective function value.

Thus, one has $\sum_{j=0}^{k}(1/\alpha_j)s_j = \sum_{j=0}^{k} z_j$ for all $k \geq 0$ and hence $(1/\alpha_j)s_j = z_j$ for all j.

Additionally, one necessarily has $r_j = (1/\alpha_j)z_j$ for all j at optimum. Indeed, otherwise decreasing r_j does not violate any constraint and makes constraint 1 slack. Then rescaling variables to restore tightness of constraint 1 improves the objective function. Thus, we need to solve

$$\begin{aligned} P_3^* = \text{maximize } & \sum_{j=0}^{\infty} z_j \\ \text{s.t.} & \\ & \sum_j (\alpha_j + 1 + 1/\alpha_j) z_j \leq 1 \\ & z \geq 0 \end{aligned} \tag{3.11}$$

■

But P_3^* is easy to analyze: there exists an optimum solution that sets all z_j to zero except for a j that minimizes $(\alpha_j + 1 + 1/\alpha_j)$. For all non-negative x, $f(x) = 1 + x + 1/x$ is minimized when $x = 1$, and $f(1) = 3$. This gives $P_3^* \leq 1/3$, and hence $\Delta' \leq 1/3$, or $ALG' \geq (2/3)OPT'$. Thus, we have proved

Theorem 35 *For any bipartite graph $G_1 = (P, Q, E_1)$ there exists a subforest G'_1 of G such that for any graph $G_2 = (P, Q, E_2)$ the maximum matching in $G'_1 \cup G_2$ is a 2/3-approximation of the maximum matching in $G_1 \cup G_2$; further, it suffices to choose G'_1 to be the matching skeleton of G_1.*

Corollary 36 $CC(\frac{1}{3}, n) = O(n)$.

Theorem 35 also implies that the matching skelton gives a linear size 1/2-cover of G.

Corollary 37 *For any bipartite graph $G = (P, Q, E)$, the matching skeleton G' is a $\frac{1}{2}$-cover of G.*

Proof: We need to show that for any $A \subseteq P, B \subseteq Q, |A|, |B| > n/2$ such that there exists a perfect matching between A and B in G one has $E' \cap (A \times B) \neq \emptyset$. Let $G_2 = (P \cup P', Q \cup Q', M_P \cup M_Q)$ be a graph that consists of a perfect matching from a new set of vertices P' to $Q \setminus B$ and a matching from a new set of vertices Q' to $P \setminus A$. Then the maximum matching in $G \cup G_2$ is of size $(3/2)n$.

By the max-flow min-cut theorem, the size of the matching in $G' \cup G_2$ is no larger than $|P \setminus A| + |Q \setminus B| + |E' \cap (A \times B)|$. By Theorem 35 the approximation ratio is at least 2/3, and $|P \setminus A| + |Q \setminus B| < n$, so it must be that $|E' \cap (A \times B)| > 0$. ■

3.4 $O(n)$ communication protocol for $CC_v(\frac{1}{4}, n)$

In this section we prove that $CC_v(\epsilon, n) = O(n)$ for all $\epsilon < 1/4$. In particular, we show that given a bipartite graph $G_1 = (P_1, Q, E_1)$, there exists a forest $F \subseteq E_1$ such that for any $G_2 = (P_2, Q, E)$ that may share nodes on the Q side with G_1 but not on the P side, the maximum matching in $G_1' \cup G_2$ is a 3/4-approximation of the maximum matching in $G_1 \cup G_2$. The broad outline of the proof is similar to the previous section, but we can now assume a special optimal matching using the assumption that G_2 may only share nodes with G_1 on the Q side.

We first prove

Lemma 38 *Let $G = (P, Q, E)$ be a bipartite graph and let $S \subseteq P$ be such that $|\Gamma(U)| \geq |U|$ for all $U \subseteq S$. Then there exists a maximum matching in G that matches all vertices of S.*

Proof: Let M be a maximum matching in $G_1 \cup G_2$ that leaves a nonempty set $U \subseteq S$ of vertices exposed. Let U be the largest subset of S exposed by M. We will show how to obtain a different maximum matching M' that leaves one fewer nodes exposed. Orient edges of the matching M from Q to P and orient all other edges from P to Q. Denote the set of all nodes reachable from U by $\Gamma^*(U)$. Suppose that no node outside S is reachable in this directed graph. Then we have $|\Gamma^*(U) \cap Q| = |\Gamma^*(U) \cap P| - |U|$, a contradiction since

1. $\Gamma^*(U) \cap P \subseteq S$ by assumption;
2. $\Gamma^*(U) \cap Q = \Gamma(\Gamma^*(U) \cap P)$.

Thus, there exists an (even length) path in this directed graph from U to $P \setminus S$. Swapping edges in and out of M along this path decreases the number of unmatched nodes in S by one while preserving the size of the matching. Repeating the argument, we obtain a maximum matching in $G_1 \cup G_2$ that matches all of S.

■

We also need

Lemma 39 *Let $G_1 = (P_1, Q, E), G_2 = (P_2, Q, E)$ and let G_1' be the matching skeleton of G_1. Let $(A_1 \cup B_1, A_2 \cup B_2)$ be a saturating cut corresponding to a maximum matching in $G_1' \cup G_2$. Then,*

1. *for all $j < 0$ one has $S_j \cap B_2 = \emptyset$;*
2. *for all $j \geq 0$ one has $|\Gamma_{B_1}(S_j \cap A_1)| \geq (1/\alpha_j)|S_j \cap A_1|$*
3. *for all edges $e = (u, v) \in (A_1 \times B_2) \cap E_1$ one has $u \in S_j$ for some $j \geq 0$ and $v \in T_i$ for some i, $0 \leq i \leq j$.*

Proof: We start by showing part (1) of the lemma. By the choice of the cut $(A_1 \cup B_1, A_2 \cup B_2)$ all of A_2 can be matched to $Q \setminus B_1$ in $G_1' \cup G_2$. Let $T^* = \Gamma_{G_1'}(S_j \cap B_2)$. One has $|T^*| \geq (1/\alpha_j)|S_j \cap B_2|$. Hence, since vertices only arrive on the P side, one has $|\Gamma_{G_1' \cup G_2}(T^*) \setminus B_1| \leq \alpha_j |T^*| < |T^*|$, which contradicts the choice of the cut $(A_1 \cup B_1, A_2 \cup B_2)$.

Part (2) follows directly by Lemma 31 together with the assumption that $(A_1 \times B_2) \cap E = \emptyset$. Now (3) follows from (1) together with the fact that edges $e \in E_1 \setminus E_1'$ that have one endpoint in $T_i, i \geq 0$ can only go to S_j for some $j \geq 0$ by construction of G_1'. ■

We now prove the main theorem of this section:

Theorem 40 *Let $G_1 = (P_1, Q, E_1), G_2 = (P_2, Q, E_2)$ be bipartite graphs that share the vertex set on one side. Let G_1' be the matching skeleton of G_1. Then the maximum matching in $G_1' \cup G_2$ is a $3/4$-approximation of the maximum matching in $G_1 \cup G_2$.*

Proof: Let $(S_j, T_j), j = -\infty, \ldots, +\infty$ be the pairs from the definition of G'. Consider a saturating cut $(A_1 \cup B_1, A_2 \cup B_2)$ in $G_1' \cup G_2$. Recall that $A_1, A_2 \subseteq P_1 \cup P_2, B_1, B_2 \subseteq Q$, $(A_1 \times B_2) \cap (E_1' \cup E_2) = \emptyset$, $ALG = |B_1| + |A_2|$.

Let $S := \bigcup_{j \geq 0} S_j$. Choose a maximum matching M in $G_1 \cup G_2$ such that M matches all of S, as guaranteed by Lemma 38. Define

$$K_j = \{v \in \Gamma_{G_1'}(S_j) \cap B_2 : M(v) \notin S\}$$
$$K_j^* = \{v \in \Gamma_{G_1'}(S_j) \cap B_1 : M(v) \notin S\}$$

By Lemma 39 there are no edges in G_1 from $T_j, j < 0$ to B_2. This implies that

$$((A_1 \setminus S) \times B_2) \cap (E_1 \cup E_2) = \emptyset. \tag{3.12}$$

This allows us to obtain the following bound on the size of the matching M, which we denote by OPT. It follows from 3.12 that a matching edge that has an endpoint in $A_1 \setminus S$ necessarily has the other endpoint either in K_j^* for some j or in $B_1 \setminus \Gamma_{G_1'}(S)$. Hence, we have

$$OPT \leq |S| + \sum_{j \geq 0} (|K_j| + |K_j^*|) + (|B_1 \setminus \Gamma_{G_1'}(S)| + |A_2 \setminus (S \cup \bigcup_{j \geq 0} M(K_j))|). \tag{3.13}$$

Indeed, if an edge $\epsilon \in M$ has an endpoint in S, it is counted by the first term. Otherwise if e has an endpoint in $\Gamma_{G_1'}(S_j) \cap B_2$ for some j, it is counted in K_j; if e has an endpoint in $\Gamma_{G_1'}(S_j) \cap B_1$

for some j, it is counted in K_j^*. Finally, if e satisfies none of the above conditions, it must have one endpoint in either $B_1 \setminus \Gamma_{G_1'}(S)$ or $A_2 \setminus (S \cup \bigcup_{j\geq 0} M(K_j))$ by 3.12. Note that an edge $e \in M$ may satisfy more than one of these conditions, and hence we are only getting an upper bound on OPT.

By definition of the cut $(A_1 \cup B_1, A_2 \cup B_2)$ we also have

$$\begin{aligned} ALG =& |B_1| + |A_2| = |S \cap A_2| + \sum_{j\geq 0} |M(K_j)| + |\Gamma_{G_1'}(S) \cap B_1| \\ &+ (|B_1 \setminus \Gamma_{G_1'}(S)| + |A_2 \setminus (S \cup \bigcup_{j\geq 0} M(K_j))|), \end{aligned} \tag{3.14}$$

where we use the fact that $M(K_j) \subseteq A_2 \setminus S$ by definition of K_j together with 3.12. Thus, since $|M(K_j)| = |K_j|$, it is sufficient to show that

$$|S \cap A_2| + \sum_{j\geq 0} |K_j| + |\Gamma_{G_1'}(S) \cap B_1| \geq (3/4)(|S| + \sum_{j\geq 0} |K_j| + |K_j^*|)$$

Let

$$\begin{aligned} ALG' &= |S \cap A_2| + \sum_{j\geq 0} |K_j| + |\Gamma_{G_1'}(S) \cap B_1| \\ OPT' &= |S| + \sum_{j\geq 0} |K_j| + |K_j^*|. \end{aligned}$$

Let $x_j := |S_j|, z_j = |S_j \cap A_1|, w_j := |\Gamma_{G_1'}(S_j \cap A_1)|, r_j^* := |K_j^*|, r_j := |K_j|$.

We will derive relations between these variables using the properties of the matching skeleton. By construction of G_1' we have

$$(S_i \times T_j) \cap E_1 = \emptyset, \forall i < j. \tag{3.15}$$

Define *canonical cuts* (U_k, W_k) as

$$U_k = \bigcup_{j=0}^{k} S_j \subseteq P_1, W_k = \bigcup_{j=0}^{k} T_j \subseteq Q. \tag{3.16}$$

By (3.15) we have that $(U_k \times (Q \setminus W_k)) \cap (E_1 \cup E_2) = \emptyset$.

Since M matches all of S, we have using the fact that canonical cuts are empty that for each $k \geq 0$

$$|U_k| \leq |W_k| - \sum_{j=0}^{k} (|K_j| + |K_j^*|).$$

Since $|T_j| = \alpha_j |S_j|$ by definition of G_1' and since T_j are disjoint, this can be equivalently stated in terms of the new variables as

$$\sum_{j=0}^{k} ((1/\alpha_j) x_j - r_j - r_j^*) \geq \sum_{j=0}^{k} x_j, \forall k \geq 0. \tag{3.17}$$

Thus, in terms of the new variables we have

$$OPT' = \sum_{j=0}^{\infty} x_j + \sum_{j=0}^{\infty} r_j + \sum_{j=0}^{\infty} r_j^*. \tag{3.18}$$

Similarly,

$$ALG' = \sum_{j=0}^{\infty} (x_j - z_j) + \sum_{j=0}^{\infty} w_j + \sum_{j=0}^{\infty} r_j \tag{3.19}$$

By Lemma 39, (3), we have $|\Gamma_{B_1}(S_j \cap A_1)| = w_j \geq (1/\alpha_j) z_j$.

Thus, putting (3.18), (3.19), (3.17) together, we have that it is sufficient to lower bound the solution of 3.20, obtaining a lower bound of P_1^* on the ratio ALG'/OPT', and hence on ALG/OPT.

$$\begin{aligned} P_1^* = \text{minimize } & \sum_{j=0}^{\infty} (x_j - z_j) + w_j + r_j \\ \text{s.t.} & \\ & \sum_{j=0}^{\infty} (x_j + r_j + r_j^*) \geq 1 \\ & \sum_{j=0}^{k} ((1/\alpha_j) x_j - r_j - r_j^*) \geq \sum_{j=0}^{k} x_j, \forall k \\ & r_j^* \leq w_j \\ & w_j \geq (1/\alpha_j) z_j \\ & x, z, w, r, r^* \geq 0 \end{aligned} \tag{3.20}$$

We now transform 3.20 in two steps to obtain bounds $P_3^* \leq P_2^* \leq P_1^*$, and then show that $P_3^* \geq 3/4$.

First note that at the optimum one has $r \equiv 0$ since decreasing r and scaling all variables appropriately does not violate any constraints and only improves the solution. Next, we show that at the optimum, the third constraint is necessarily tight for all k. Otherwise let k be such that the constraint is not tight and let k^* be the smallest such that $k^* > k$ and $r_{k^*} > 0$.

Let

$$\begin{aligned} x' &= x \\ r^{*\prime} &= r^* + \delta e_k - \delta e_{k^*} \\ w' &= w + \delta e_k - \delta e_{k^*} \\ z' &= z + \alpha_k e_k - \alpha_{k^*} e_{k^*}. \end{aligned}$$

Note that $x', r^{*\prime}, w', z'$ form a feasible solution if $\delta > 0$ is sufficiently small. Finally,

$$\sum_{j=0}^{\infty}(x'_j - z'_j) + w'_j = \left(\sum_{j=0}^{\infty}(x_j - z_j) + w_j\right) + \delta(-\alpha_k + \alpha_{k^*}) < \sum_{j=0}^{\infty}(x_j - z_j) + w_j.$$

Also, for fixed r^*, x one can maximize z_j pointwise, so $r^*_j = (1/\alpha_j)z_j$ for all j.

Thus, we have $P^*_2 \leq P^*_1$, where

$$\begin{aligned} P^*_2 = \text{minimize } & 1 - \sum_{j=0}^{\infty} z_j \\ \text{s.t.} & \\ & \sum_{j=0}^{\infty}(x_j + (1/\alpha_j)z_j) = 1 \\ & \sum_{j=0}^{k}(1/\alpha_j - 1)x_j \geq \sum_{j=0}^{k}(1/\alpha_j)z_j, \forall k \\ & x, z \geq 0 \end{aligned} \tag{3.21}$$

Finally, we show that constraints in line 2 are necessarily tight at the optimum. Otherwise let k^* be the smallest such that constraint 2 is slack. Note that we necessarily have $x_{k^*} > 0$. Let

$$x' = x - \delta e_{k^*} + \delta e_{k^*+1}\frac{1/\alpha_{k^*} - 1}{1/\alpha_{k^*+1} - 1},$$

which is feasible for sufficiently small $\delta > 0$ and makes constraint 2 satisfied for all k. Let

$$\gamma = \sum_{j=0}^{\infty}(x_j + \alpha_j z_j) = 1 - \delta\left(1 - \frac{1/\alpha_{k^*} - 1}{1/\alpha_{k^*+1} - 1}\right) = 1 - \delta\frac{1/\alpha_{k^*+1} - 1/\alpha_{k^*}}{1/\alpha_{k^*+1} - 1} < 1.$$

Now $x'' = x'/\gamma, z'' = z/\gamma$ are feasible solutions that improve the objective function.

Thus, we have $x_j = z_j/(1 - \alpha_j)$ for all $j > 0$ (note that $x_0 = 0$ at the optimum for the same reason as $r \equiv 0$). Thus, we get $P^*_3 \leq P^*_2$, where

$$\begin{aligned} P^*_3 = \text{minimize } & 1 - \sum_{j=1}^{\infty} z_j \\ \text{s.t.} & \\ & \sum_{j=1}^{\infty}(1/(1 - \alpha_j) + 1/\alpha_j)z_j = 1 \\ & z \geq 0 \end{aligned} \tag{3.22}$$

In order to lower bound P^*_3, it is sufficient to minimize $f(\alpha) = 1/(1-\alpha) + 1/\alpha$ over all $\alpha \in (0, 1]$.

One has $f'(\alpha) = 1/(1-\alpha)^2 - 1/\alpha^2, f'(1/2) = 0$ and $f''(\alpha) = 2/(1-\alpha)^3 + 2/(1-\alpha)^3 > 0$. Hence, the unique minimum is attained at $\alpha = 1/2$.

Thus, we have $z_j = 1/4$ for $\alpha_j = 1/2$ and zero otherwise. The objective value is $3/4$, proving that $3/4 \leq P_3^* \leq P_2^* \leq P_1^*$, and hence $ALG/OPT \geq 3/4$. ■

3.5 One-pass streaming with vertex arrivals

Let $G_i = (P_i, Q, E_i)$ be a sequence of bipartite graphs, where $P_i \cap P_j = \emptyset$ for $i \neq j$. For a graph G, we denote by SPARSIFY$^*(G)$ the matching skeleton of G modified as follows: for each pair $(S_j, T_j), j < 0$ keep an arbitrary matching of S_j to a subset of T_j, discarding all other edges, and collect all these matchings into the (S_0, T_0) pair. Note that we have $S_j \subseteq P$, where P is the side of the graph that arrives in the stream. We have

Lemma 41 *Let $G = (P, Q, E)$ be a bipartite graph. Let $G' = SPARSIFY^*(G)$. Let $(S_j, T_j), j = 0, \ldots, +\infty$ denote the set of expanding pairs. Then $E \cap (S_i \times T_j) = \emptyset$ for all $i < j$.*

Let

$$G'_1 = \text{SPARSIFY}^*(G_1), \text{ and } G'_i = \text{SPARSIFY}^*(G'_{i-1} \cup G_i). \tag{3.23}$$

We will show that for each $\tau > 0$ the maximum matching in G'_τ is at least a $1 - 1/e$ fraction of the maximum matching in $\bigcup_{i=1}^{\tau} G_i$. We will slightly abuse notation by denoting the set of expanding pairs in G'_τ by $(S_\alpha(\tau), T_\alpha(\tau))$. Recall that we have $\alpha \in (0, 1]$, and $|S_\alpha(\tau)| = \alpha |T_\alpha(\tau)|$. We need the following

Definition 42 *For a vertex $u \in P$ define its* level *after time τ, denoted by $\alpha_u(\tau)$, as the value of α such that $u \in S_\alpha(\tau)$. Similarly, for a vertex $v \in Q$ define its* level *after time τ, denoted by $\alpha_v(\tau)$, as the value of α such that $u \in T_\alpha(\tau)$. Note that for a vertex u is at level $\alpha = \alpha_u(\tau)$ the expansion of the pair $(S_\alpha(\tau), T_\alpha(\tau))$ that it belongs to is $1/\alpha$.*

Before describing the formal proof, we give an outline of the main ideas. In our analysis, we track the structure of the matching skeleton maintained by the algorithm over time. For the purposes of our analysis, at each time τ, every vertex is characterized by two numbers: its *initial level* β when it first appeared in the stream and its *current level* α at time τ (we denote the set of such vertices at time τ by $S_{\alpha,\beta}(\tau)$). Informally, we first deduce that the matching edges that our algorithm misses may only connect a vertex in $S_{\alpha,\beta}(\tau)$ to a vertex in $T_{\beta'}(\tau)$ for $\beta' \geq \beta$, and hence we are interested in the distribution of vertices among the sets $S_{\alpha,\beta}(\tau)$. We show that vertices that initially appeared at lower levels and then migrated to higher levels are essentially the most detrimental to the approximation ratio. However, we prove that for every $\lambda \in (0, 1]$, which can be thought of as a 'barrier', the number of vertices that initially appeared at level $\beta < \lambda$ but migrated to a level $\alpha \geq \lambda$ can never be larger than $\lambda \left| \bigcup_{\gamma \in [\lambda, 1]} T_\gamma(\tau) \right|$ at any time τ. This leads to a linear program whose optimum lower bounds the approximation ratio, and yields the $(1 - 1/e)$ approximation guarantee.

Lemma 43 *For all $u \in P$ and for all τ, $\alpha_u(\tau+1) \geq \alpha_u(\tau)$. Similarly for $v \in Q$, $\alpha_v(\tau+1) \geq \alpha_v(\tau)$.*

Proof: We prove the statement by contradiction. Let τ be the smallest such that $\exists \alpha \in (0,1]$ such that $R := \{u \in P : u \in S_\alpha(\tau), \alpha_u(\tau+1) < \alpha_u(\tau)\} \neq \emptyset$. Let $\alpha^* = \min_{u \in R} \alpha_u(\tau+1)$ (we have $\alpha^* < \alpha$ by assumption). Let $R^* = R \cap S_{\alpha^*}(\tau+1)$. Note that $R^* \subseteq S_\alpha(\tau)$. We have

$$|\Gamma_{G'_\tau}(R^*)| \geq |\Gamma_{G'_{\tau+1}}(R^*)| \geq (1/\alpha^*)|R^*| > (1/\alpha)|R^*|. \tag{3.24}$$

Since $|\Gamma_{G'_\tau}(S_\alpha(\tau))| = (1/\alpha)|S_\alpha(\tau)|$, (3.24) implies that $S_\alpha(\tau) \setminus R^* \neq \emptyset$. However, since $|\Gamma_{G'_\tau}(S_\alpha(\tau) \setminus R^*)| \geq (1/\alpha)|S_\alpha(\tau) \setminus R^*|$, one has

$$\Gamma_{G'_\tau}(S_\alpha(\tau) \setminus R^*) \cap \Gamma_{G'_\tau}(R^*) \neq \emptyset.$$

This, however, contradicts the assumption that $(S_\alpha(\tau) \setminus R^*) \cap S_{\alpha^*}(\tau+1) = \emptyset$ and the fact that $G'_{\tau+1} = \text{SPARSIFY}^*(G'_\tau, G_{\tau+1})$.

The same argument also proves the monotonicity of levels for $v \in Q$. ■

Let $S_{\alpha,\beta}(\tau)$ denote the set of vertices in $u \in P$ such that

1. $u \in S_\beta(\tau')$, where τ' is the time when u arrived (i.e. $u \in P_{\tau'}$), and
2. $u \in S_\alpha(\tau)$.

Note that one necessarily has $\alpha \geq \beta$ by Lemma 43 for all nonempty $S_{\alpha,\beta}$. We will need the following

Lemma 44 *For all τ one has for all $\lambda \in (0,1]$*

$$\left((Q \setminus \bigcup_{\alpha \in [\lambda,1]} T_\alpha(\tau)) \times \bigcup_{\beta \in [\lambda,1]} S_{\alpha,\beta}(\tau) \right) \cap \bigcup_{t=1}^{\tau} E_t = \emptyset.$$

Proof: A vertex $u \in S_{\alpha,\beta}(\tau)$ with $\beta \geq \lambda$ that arrived at time τ_u could only have edges to $v \in T_{\lambda'}(\tau_u)$ for $\lambda' \geq \lambda$. By Lemma 43, such vertices v can only belong to $T_{\lambda''}(\tau)$ for some $\lambda'' \geq \lambda' \geq \beta \geq \lambda$, and the conclusion follows with the help of Lemma 41. ■

Let $t_\alpha(\tau) = |T_\alpha(\tau)|, s_{\alpha,\beta}(\tau) = |S_{\alpha,\beta}(\tau)|$. The quantities $t_\alpha(\tau), s_{\alpha,\beta}(\tau)$ are defined for $\alpha, \beta \in D = \{\Delta k : 0 < k \leq 1/\Delta\}$, where $1/\Delta$ is a sufficiently large integer (note that all relevant values of α, β are rational with denominators bounded by n). In what follows all summations over levels are assumed to be over the set D. Then

Lemma 45 *For all τ and for all $\alpha \in (0,1]$, the quantities $t_\alpha(\tau), s_{\alpha,\beta}(\tau)$ satisfy*

$$\sum_{\beta \in [\alpha,1]} \sum_{\delta \in (0, \alpha-\Delta]} s_{\beta,\delta}(\tau) \leq (\alpha - \Delta) \sum_{\beta \in [\alpha,1]} t_\beta(\tau). \tag{3.25}$$

Proof: The proof is by induction on τ.

Base: $\tau = 0$ At $\tau = 0$ the lhs is zero, so the relation is satisfied.

Inductive step: $\tau \to \tau + 1$ Fix $\alpha \in (0, 1)$. For all $\gamma \in (0, \alpha - \Delta]$ let

$$R_\gamma(\tau) = S_\gamma(\tau) \cap \left(\bigcup_{\beta \in [\alpha, 1]} S_\beta(\tau + 1) \right).$$

We have $|\Gamma_{G'_\tau}(R_\gamma(\tau))| \geq (1/\gamma)|R_\gamma(\tau)|$ and $\Gamma_{G'_\tau}(R_\gamma(\tau)) \subseteq \bigcup_{\beta \in [\alpha,1]} T_\beta(\tau + 1)$.
Also, we have by Lemma 43 that

$$\left(\bigcup_{\beta \in [\alpha,1]} T_\beta(\tau) \right) \cup \left(\bigcup_{\gamma \in (0, \alpha - \Delta]} \Gamma_{G'_\tau}(R_\gamma(\tau)) \right) \subseteq \bigcup_{\beta \in [\alpha,1]} T_\beta(\tau + 1).$$

Moreover, since $\Gamma_{G'_\tau}(R_\gamma(\tau))$ are disjoint for different γ and disjoint from $T_\beta(\tau), \beta \in [\alpha, 1]$, letting $r_\gamma(\tau) = |R_\gamma(\tau)|$, we have

$$\sum_{\beta \in [\alpha,1]} t_\beta(\tau + 1) \geq \sum_{\beta \in [\alpha,1]} t_\beta(\tau) + \sum_{\gamma \in (0, \alpha - \Delta]} \frac{1}{\gamma} r_\gamma(\tau) \geq \sum_{\beta \in [\alpha,1]} t_\beta(\tau) + \frac{1}{\alpha - \Delta} \sum_{\gamma \in (0, \alpha - \Delta]} r_\gamma(\tau). \tag{3.26}$$

Furthermore, by Lemma 43

$$\sum_{\beta \in [\alpha,1]} \sum_{\delta \in (0, \alpha - \Delta]} s_{\beta,\delta}(\tau + 1) = \sum_{\beta \in [\alpha,1]} \sum_{\delta \in (0, \alpha - \Delta]} s_{\beta,\delta}(\tau) + \sum_{\gamma \in (0, \alpha - \Delta]} r_\gamma(\tau) \tag{3.27}$$

Since by inductive hypothesis

$$\sum_{\beta \in [\alpha,1]} t_\beta(\tau) \geq \frac{1}{\alpha - \Delta} \sum_{\beta \in [\alpha,1]} \sum_{\delta \in (0, \alpha - \Delta]} s_{\beta,\delta}(\tau). \tag{3.28}$$

we have by combining (3.26), (3.27) and (3.28)

$$\begin{aligned}
\sum_{\beta \in [\alpha,1]} t_\beta(\tau + 1) &\geq \sum_{\beta \in [\alpha,1]} t_\beta(\tau) + \frac{1}{\alpha - \Delta} \sum_{\gamma \in (0, \alpha - \Delta]} r_\gamma(\tau) \\
&\geq \frac{1}{\alpha - \Delta} \sum_{\beta \in [\alpha,1]} \sum_{\delta \in (0, \alpha - \Delta]} s_{\beta,\delta}(\tau) \\
&\quad + \frac{1}{\alpha - \Delta} \sum_{\beta \in [\alpha,1]} \sum_{\delta \in (0, \alpha - \Delta]} (s_{\beta,\delta}(\tau + 1) - s_{\beta,\delta}(\tau)) \\
&= \frac{1}{\alpha - \Delta} \sum_{\beta \in [\alpha,1]} \sum_{\delta \in (0, \alpha - \Delta]} s_{\beta,\delta}(\tau + 1).
\end{aligned}$$

■

In what follows we only consider sets $S_{\alpha,\beta}(\tau), T_\alpha(\tau)$ for fixed τ, and omit τ for brevity. Let $S = \bigcup_{\alpha,\beta} S_{\alpha,\beta}$. Choose a maximum matching M in G_τ that matches all of S, as guaranteed by Lemma 38. Let γ denote the number of vertices in T_1 that are matched outside of S by M (note that no vertices of $T_\alpha, \alpha \in (0,1)$ are matched outside of S by lemma 44). For each $\alpha \in (0,1]$ let $r_\alpha \leq t_\alpha$ denote the number of vertices in T_α that are not matched by M. Then the following is immediate from lemma 44.

Lemma 46 *For all $\lambda \leq 1$*

$$\sum_{\alpha\in[\lambda,1]} t_\alpha \geq \sum_{\alpha\in[\lambda,1],\beta\in[\lambda,1]} s_{\alpha,\beta} + \sum_{\alpha\in[\lambda,1]} r_\alpha + \gamma. \tag{3.29}$$

Proof: Follows from Lemma 44. ■

We also have

$$\sum_{\beta\in[\alpha,1]} \sum_{\delta\in(0,1]} s_{\beta,\delta} = \sum_{\beta\in[\alpha,1]} \beta t_\beta \tag{3.30}$$

for all $\alpha \in (0,1]$.

By Lemma 45 and Lemma 46, we get

$$\begin{aligned} ALG &= \sum_{\alpha\in(0,1)} (t_\alpha - r_\alpha) + (t_1 - r_1 - \gamma) \\ OPT &= ALG + \gamma \\ t_1 &\geq \gamma + r_1. \end{aligned}$$

Thus, we need to minimize ALG/OPT subject to $t_1 \geq r_1 + \gamma, t_\alpha, s_{\alpha,\beta} \geq 0$ and

$$\begin{aligned} &\forall \alpha \in (0,1] : \sum_{\beta\in[\alpha,1]} t_\beta \geq \gamma + \sum_{\beta\in[\alpha,1]} \sum_{\delta\in[\alpha,1]} s_{\beta,\delta} + \sum_{\beta\in[\alpha,1]} r_\beta. \\ &\forall \alpha \in (0,1] : \sum_{\beta\in[\alpha,1]} \sum_{\delta\in(0,\alpha-\Delta]} s_{\beta,\delta} \leq (\alpha - \Delta) \sum_{\beta\in[\alpha,1]} t_\beta \\ &\forall \alpha \in (0,1] \sum_{\beta\in[\alpha,1]} \sum_{\delta\in(0,1]} s_{\beta,\delta} = \sum_{\beta\in[\alpha,1]} \beta t_\beta. \end{aligned} \tag{3.31}$$

We start by simplifying (3.31). First note that we can assume without loss of generality that $r_1 = 0$. Indeed, if $r_1 > 0$, we can decrease r_1 to 0 and increase γ to keep ALG constant, without violating any constraints, only increasing OPT. Furthermore, we have wlog that $t_1 > 0$ since otherwise $ALG/OPT = 1$. Finally, note that setting $t_1 = \gamma$ only makes the ratio ALG/OPT smaller, so it is sufficient to lower bound $\sum_{\alpha\in(0,1)}(t_\alpha - r_\alpha)$ in terms of γ, and for this purpose we can set $\gamma = 1$ since this only fixes the scaling of all variables. Thus, it is sufficient to lower bound the optimum of (3.32), obtaining a lower bound of $\frac{P_1^*}{P_1^*+1}$ on the ratio ALG/OPT.

$$
\begin{aligned}
P_1^* = \text{minimize} \quad & \sum_{\alpha\in(0,1)} (t_\alpha - r_\alpha) \\
\text{s.t.} \quad & \\
\forall \alpha \in (0,1] : \quad & \sum_{\beta\in[\alpha,1]} t_\beta \geq 1 + \sum_{\beta\in[\alpha,1]} \sum_{\delta\in[\alpha,1]} s_{\beta,\delta} + \sum_{\beta\in[\alpha,1]} r_\beta. \\
\forall \alpha \in (0,1] : \quad & \sum_{\beta\in[\alpha,1]} \sum_{\delta\in(0,\alpha-\Delta]} s_{\beta,\delta} \leq (\alpha - \Delta) \sum_{\beta\in[\alpha,1]} t_\beta \\
\forall \alpha \in (0,1] \quad & \sum_{\beta\in[\alpha,1]} \sum_{\delta\in(0,1]} s_{\beta,\delta} = \sum_{\beta\in[\alpha,1]} \beta t_\beta \\
& t_\alpha, s_{\alpha,\beta} \geq 0.
\end{aligned}
\tag{3.32}
$$

Combining constraints 2 and 3 of (3.32), we get

$$\sum_{\beta=\alpha}^{1} (1 + \alpha - \Delta) t_\beta \geq \gamma + \sum_{\beta=\alpha}^{1} \beta t_\beta.$$

Thus, it is sufficient to lower bound the optimum of

$$
\begin{aligned}
P_2^* = \text{minimize} \quad & \sum_{\alpha\in(0,1)} (t_\alpha - r_\alpha) \\
\text{s.t.} \quad & \\
\forall \alpha \in (0,1] : \quad & \sum_{\beta\geq\alpha} (1 - \beta + \alpha - \Delta) t_\beta \geq 1 + \sum_{\beta\in[\alpha,1)} r_\alpha. \\
& t_\alpha \geq 0.
\end{aligned}
\tag{3.33}
$$

We first show that one has $r_\alpha = 0$ for all $\alpha \in [0,1)$ at the optimum. Indeed, suppose that $r_{\alpha^*} > 0$ for some $\alpha^* \in (0,1)$. Then since the coefficient of t_{α^*} is $(1 - \alpha^* + \alpha - \Delta) \leq 1 - \Delta < 1$, $\beta = \alpha^* \geq \alpha$, we can decrease r_α^* by some $\delta > 0$ and also decrease t_{α^*} by $\frac{\delta}{1-\Delta} < \delta$, keeping all constraints satisfied and improving the value of the objective function.

Thus, we arrive at the final LP, whose optimum we need to lower bound:

$$
\begin{aligned}
P_3^* = \text{minimize} \quad & \sum_{\alpha\in(0,1)} t_\alpha \\
\text{s.t.} \quad & \\
\forall \alpha \in (0,1] : \quad & \sum_{\beta\geq\alpha} (1 - \beta + \alpha - \Delta) t_\beta \geq 1. \\
& t_\alpha \geq 0.
\end{aligned}
\tag{3.34}
$$

We now show that all constraints are necessarily tight at the optimum. Let $\alpha^* \in [0,1]$ be the largest such that constraint 1 is not tight. Note that one necessarily has $t_{\alpha^*} > 0$. Let

$$t' = t - \delta e_{\alpha^*} + \frac{\delta}{1+\Delta} e_{\alpha^* - \Delta}.$$

We now verify that all constrains are satisfied. For $\alpha > \alpha^*$ all constraints are satisfied since we did not change t. For $\alpha = \alpha^*$, the constraint is satisfied since it was slack for t and δ is sufficiently small.

For $\alpha < \alpha^*$, i.e. $\alpha \leq \alpha^* - \Delta$ since we are considering only $\alpha \in D$, we have

$$\begin{aligned}\sum_{\beta \geq \alpha} (1 - \beta + \alpha - \Delta) t'_\beta &= \sum_{\beta \geq \alpha} (1 - \beta + \alpha - \Delta) t_\beta + \delta \left(\frac{1 - (\alpha^* - \Delta) + \alpha - \Delta}{1+\Delta} - (1 - \alpha^* + \alpha - \Delta) \right) \\ &= \sum_{\beta \geq \alpha} (1 - \beta + \alpha - \Delta) t_\beta + \frac{\delta \Delta (\alpha^* - \alpha - \Delta)}{1+\Delta} \geq \sum_{\beta \geq \alpha} (1 - \beta + \alpha - \Delta) t_\beta \geq 1.\end{aligned}$$

Thus, at the optimum we have

$$\sum_{\beta \geq \alpha} (1 + (\alpha - \beta - \Delta)) t_\beta = 1, \forall \alpha \in [0,1]. \tag{3.35}$$

Subtracting (3.35) for $\alpha + \Delta$ from (3.35) for α, we get

$$\begin{aligned}\sum_{\beta \geq \alpha} (1 + (\alpha - \beta - \Delta)) t_\beta - \sum_{\beta \geq \alpha + \Delta} (1 + (\alpha + \Delta - \beta - \Delta) t_\beta \\ = t_\alpha - \Delta \sum_{\beta \geq \alpha} t_\beta = 0.\end{aligned} \tag{3.36}$$

In other words,

$$t_\alpha = \Delta \sum_{\beta \geq \alpha} t_\beta, t_1 \geq 1. \tag{3.37}$$

Let $\delta = \frac{\Delta}{1-\Delta}$. We now prove by induction that $t_{1-k\Delta} = \delta(1+\delta)^{k-1}$ for all $k > 0$.

Base:$k = 1$ $t_{1-\Delta} = \frac{\Delta}{1-\Delta} = \delta$.

Inductive step: $k \to k+1$

$$t_{1-(k+1)\Delta} = \Delta \left(t_{1-(k+1)\Delta} + 1 + \delta \sum_{j=1}^{k} (1+\delta)^{j-1} \right)$$

Thus,

$$t_{1-(k+1)\Delta} = \delta \left(1 + \delta \sum_{j=1}^{k} (1+\delta)^{j-1} \right) = \delta \left(1 + \delta \frac{1 - (1+\delta)^k}{1 - (1+\delta)} \right) = \delta (1+\delta)^k.$$

Hence, one has

$$\sum_{\alpha\in[0,1)} t_\alpha \geq \delta \sum_{j=1}^{1/\Delta} (1+\delta)^{j-1} = \delta \frac{1-(1+\delta)^{1/\Delta}}{1-(1+\delta)} = (1+\delta)^{1/\Delta} - 1 = \left(1+\frac{\Delta}{1-\Delta}\right)^{1/\Delta} - 1 = (1-\Delta)^{-1/\Delta} - 1$$

Now, the size of the matching M is bounded by

$$OPT \leq \sum_{\alpha\in[0,1)} t_\alpha + 1.$$

On the other hand,

$$ALG \geq \sum_{\alpha\in[0,1)} t_\alpha.$$

Thus, we get

$$\frac{ALG}{OPT} = \frac{P_1^*}{P_1^*+1} = 1 - \frac{1}{P_1^*+1} \geq 1 - \frac{1}{P_3^*+1} \geq 1 - (1-\Delta)^{1/\Delta} \geq 1 - 1/e$$

since $(1-\Delta)^{1/\Delta} \leq 1/e$ for all $\Delta \geq 0$. We have now proved

Theorem 47 *There exists a deterministic $O(n)$ space 1-pass streaming algorithm for approximating the maximum matching in bipartite graphs in the vertex arrival model.*

Proof: Run the algorithm given in (3.23), letting $|P_i| = 1$, i.e. sparsifying as soon as a new vertex comes in. The algorithm only keeps a sparsifier G_i' in memory, which takes space $O(n)$. ■

3.6 Constructions of Ruzsa-Szemerédi graphs

In this section we give two extensions of constructions of Ruzsa-Szemerédi graphs from [29]. The first construction shows that for any constant $\epsilon > 0$ there exist $(1/2-\epsilon)$-Ruzsa-Szemerédi graphs with superlinear number of edges. We use this construction in section 3.7 to prove that our bound on $CC(\epsilon, n), \epsilon < 1/3$ is tight. The second construction that we present is a generalization to lop-sided graphs, which we use in section 3.7 to prove that our bound on $CC_v(\epsilon, n), \epsilon < 1/4$ is tight. Specifically, we show the following results:

Lemma 48 *For any constant $\epsilon > 0$ there exists a family of bipartite $(1/2-\epsilon)$-Ruzsa-Szemerédi graphs with $n^{1+\Omega(1/\log\log n)}$ edges.*

Lemma 49 *For any constant $\delta > 0$ there exists a family of bipartite Ruzsa-Szemerédi graphs $G = (X, Y, E)$ with $|X| = n$, $|Y| = 2n$ such that (1) the edge set E is a union of $n^{\Omega_\delta(1/\log\log n)}$ induced 2-matchings $M_1, \ldots, M_k$ of size at least $(1/2 - O(\delta))|X|$, and (2) for any $j \in [1:k]$ the graph G contains a matching M_j^* of size at least $(1 - O(\delta))|X|$ that avoids $Y \setminus (M_j \cap Y)$.*

The proofs of these results are based on an adaptation of Theorem 16 in [29] (see also [77]), which constructs *bipartite* 1/3-Ruzsa-Szemerédi graphs with superlinear number of edges. The main idea of the construction, use of a large family of nearly orthogonal vectors derived from known families of error correcting codes, is the same. A technical step is required to go from matchings of size $1/3$ to matchings of size $1/2 - \epsilon$ for any $\epsilon > 0$. Since the result does not follow directly from [29], we give a complete proof in the full version.

3.6.1 Balanced graphs

The following lemma is an adaptation of Theorem 16 in [29] (see also [77]), where *bipartite* 1/3-Ruzsa-Szemerédi graphs with a superlinear number of edges are constructed. The main idea of the construction, i.e. the use of a large family of nearly orthogonal vectors derived from known families of error correcting codes, is the same. A technical step is required to go from matchings of size $1/3$ to matchings of size $1/2 - \epsilon$ for any $\epsilon > 0$. Since the result does not follow directly from [29], we give the argument here.

Proof of Lemma 48: Let $X = Y = [m^2]^m$ for some integer $m > 0$. We will refer to vertices in X and Y as points in $[m^2]^m$. Matchings M_T will be indexed by subsets $T \subseteq [m]$.

Fix $T \subseteq [m]$. Let $L_s = \{x : \sum_{i \in T} x_i = s\}$. Define red, white and blue strips as follows. Choose $w = 2(1 + 2/\epsilon)(\epsilon m/6)$and define

$$R_k = \bigcup_{s=kw}^{kw+(2/\epsilon)(\epsilon m/6)-1} L_s$$

$$W_k = \bigcup_{s=kw+(2/\epsilon)(\epsilon m/6)}^{kw+(1+(2/\epsilon))(\epsilon m/6)-1} L_s$$

$$B_k = \bigcup_{s=kw+(1+2/\epsilon)(\epsilon m/6)}^{kw+(1+4/\epsilon)(\epsilon m/6)-1} L_s$$

$$W'_k = \bigcup_{s=kw+(1+4/\epsilon)(\epsilon m/6)}^{(k+1)w-1} L_s$$

Finally, define $B = \bigcup_k B_k, R = \bigcup_k R_k, W' = \bigcup_k W'_k, W = \bigcup_k W_k$.

For $T \subseteq [m]$ let 1_T denote the characteristic vector of T. The matching M_T is defined as follows. If a blue point $b \in B^X$ has all coordinates greater than $(2/\epsilon + 1)$, match it to the point $r = b - (2/\epsilon + 1) \cdot 1_T$ in R^Y. Note that $r \in R^Y$ by the definition of B and R.

Following [29], we first note that

Lemma 50 $|M_T| \geq (1/2 - \epsilon)n - o(n)$

Proof: The only points of B that are not matched by M_T are those in the set

$$S = \{x : \exists j \in T, x_j < (2/\epsilon + 1)v_j\}.$$

However, $|S| \leq \frac{(m/6)(2/\epsilon+1)}{m^2}|X| = \frac{(2/\epsilon+1)}{6m}|X| = o(|X|)$. Hence, we have that $|B| = (1 \pm o(1))|R|$. Similarly, we have that $|W| \leq (\epsilon/(1+\epsilon) \pm o(1))|B|$ ∎

Now let T_1, T_2 be two sets in $[m]$ of size $(\epsilon/6)m$ such that $|T_1 \cap T_2| \leq (5/2)(\epsilon/6)^2 m$. We show that no edge of M_{T_1} is induced by M_{T_2}. Let b be matched to r by T_1, i.e. $b - r = (2/\epsilon + 1)1_{T_1}$. If the edge (b, r) is induced by M_{T_2}, then one of b, r is colored blue and the other is colored red in the coloring induced by T_2. In particular, b and r are separated by a white strip. Thus,

$$\left| \sum_{i \in T_2} b_i - \sum_{i \in T_2} r_i \right| \geq (\epsilon/6)m. \tag{3.38}$$

On the other hand,

$$\begin{aligned} \left| \sum_{i \in T_2} b_i - \sum_{i \in T_2} r_i \right| = \left| \sum_{i \in T_2} (b - r)_i \right| = \left| \sum_{i \in T_2} ((2/\epsilon + 1)1_{T_1})_i \right| \\ = (2/\epsilon + 1)|T_1 \cap T_2| < (2/\epsilon + 1)(5/2)(\epsilon/6)^2 m = (5/6)(1 + \epsilon/12)(\epsilon/6)m, \end{aligned} \tag{3.39}$$

a contradiction with (3.38) for any $\epsilon \leq 1/2$.

Now it suffices to exhibit a large family $\mathcal{F}$ of subsets of $[m]$ of size $(\epsilon/6)m$ with intersection at most $(5/2)(\epsilon/6)^2$. Following [29], we obtain such a family from an error-correcting code with weight $w = (\epsilon/6)m$ and Hamming distance at least $d = 2(\epsilon/6) - (5/2)(\epsilon/6)^2$. The Gilbert-Varshamov bound yields [66], for $d \leq \frac{2w(m-w)}{m}$, a family $\mathcal{F}$ such that

$$\frac{1}{m} \log |\mathcal{F}| \geq H\left(\frac{w}{m}\right) - \frac{w}{m} H\left(\frac{d}{2w}\right) - \left(1 - \frac{w}{m}\right) H\left(\frac{d}{2(m-w)}\right) - o(1)$$

Letting $\delta = \epsilon/3$ and $\gamma = 5/4$ for convenience, we have that $w/m = \delta$ and $d/m = 2\delta(1 - \gamma\delta)$. This yields

$$\frac{1}{m} \log |\mathcal{F}| \geq H(\delta) - \delta H(1 - \gamma\delta) - (1 - \delta) H\left(\frac{\delta - \gamma\delta^2}{1 - \delta}\right) - o(1)$$

Using $H(x) = H(1 - x)$ and strict convexity of $H(x)$, we get

$$\delta H(1 - \gamma\delta) + (1 - \delta) H\left(\frac{\delta - \gamma\delta^2}{1 - \delta}\right) \leq c(\delta, \gamma) + H\left(\gamma\delta^2 + (1 - \delta)\left(\frac{\delta - \gamma\delta^2}{1 - \delta}\right)\right) = c(\delta, \gamma) + H(\delta)$$

where $c(\delta, \gamma) > 0$ whenever $\gamma \neq 1$.

Hence, setting $\gamma = 5/4$ and $\delta = \epsilon/6$ yields a family of codes with $\frac{1}{m} \log |\mathcal{F}| \geq c(\epsilon/6, 5/4) - o(1)$.

Thus, we have constructed a bipartite graph $G = (X, Y, E)$ such that $E = \bigcup_{T \in \mathcal{F}} M_T$ is a union of induced matchings of size $1/2 - \epsilon - o(1)$. The number of nodes in the graph is m^{2m} and the number of matchings is $|\mathcal{F}| = 2^{(c(\epsilon/3, 5/4) - o(1))m} = 2^{\Omega(m)}$. Thus, we get a graph on $n = m^{2m}$ nodes that is a union of $2^{\Omega(m)} = n^{\Omega_\epsilon(1/\log\log n)}$ induced matchings of size $1/2 - \epsilon$. ∎

3.6.2 Lop-sided graphs

We now extend this construction to lop-sided graphs, which will be important for showing optimality of our bound on $CC_v(\epsilon, n)$.

Lemma 51 *For any constant $\delta > 0$ there exists a family of bipartite Ruzsa-Szemerédi graphs $G = (X, Y, E)$ with $|X| = n$, $|Y| = 2n$ such that*

1. *the edge set E is a union of $n^{\Omega_\delta(1/\log\log n)}$ induced 2-matchings $M_1, \ldots, M_k$ of size at least $(1/2 - O(\delta))|X|$.*
2. *for any $j \in [1:k]$ the graph G contains a matching M_j^* of size at least $(1 - O(\delta))|X|$ that avoids $Y \setminus (M_j \cap Y)$.*

Proof: Let $X' = Y = [m^2]^m$ for some integer $m > 0$. Let X be a random subset of X' that contains each element of X' with probability $1/2$. We will refer to vertices in X and Y as points in $[m^2]^m$. The matchings M_T will be indexed by subsets $T \subseteq [m]$.

Choose $w = 2C(1 + 2/\delta)p$ for p and a constant $C > 0$ to be specified later. Fix $T \subseteq [m]$. Let $L_s = \{x : \sum_{i \in T} x_i = s + (w/2) \cdot Q_T\}$, where Q_T is a Bernoulli 0/1 random variable with on probability $1/2$. Define red, white and blue strips as follows. Define

$$R_k = \bigcup_{s=kw}^{kw+C(2/\delta)p-1} L_s$$

$$W_k = \bigcup_{s=kw+C(2/\delta)p}^{kw+C(1+(2/\delta))p-1} L_s$$

$$B_k = \bigcup_{s=kw+C(1+2/\delta)p}^{kw+C(1+4/\delta)p-1} L_s$$

$$W'_k = \bigcup_{s=kw+C(1+4/\delta)p}^{(k+1)w-1} L_s$$

Define $B = \bigcup_k B_k, R = \bigcup_k R_k, W' = \bigcup_k W'_k, W = \bigcup_k W_k$.

Here we are assuming that $\delta \in (0, 1)$ is such that $2/\delta$ is an integer.

Fix k. For two vertices $u, v \in B_k$ we say that $u \sim v$ if $u - v = \lambda(1 + 2/\delta) \cdot 1_T$ for some λ (note that since $u, v \in B_k$, we have $\lambda \in [-C, C]$). We write $S_v \subseteq Y$ to denote the equivalence class of v. Note that $|S_v| \geq C/2$ for all v. Also, let

$$T_v = \{u \in X : u = w - C(1 + 2/\delta) \cdot 1_T, w \in S_v\}.$$

Note that for any $v \in B_k$ one has $T_v \subseteq R_k$. Note that T_v is a random set (determined by the random choice of $X \subset X'$).

We now define a 2-matching from (a subset of) T_v to S_v. First note that $\mathbf{E}[|T_v|] = \frac{1}{2}|S_v|$. Furthermore, since X is obtained from X by independent sampling, the events $\{v \in X\}$ are independent

conditional on the value of Q_T. Thus, standard concentration inequalities apply (see, e.g. [41]) and we get

$$\mathbf{Pr}\left[|T_v| \notin (1\pm\delta)\frac{1}{2}|S_v|\right] \leq e^{-\delta^2(1/2)|S_v|/4} = e^{-\delta^2 C/16} \leq \delta/4$$

for $C > 16\ln(4/\delta)/\delta^2$. We now classify points $v \in B_k$ as good or bad depending on the how close $|T_v|$ is to its expectation. In particular, mark v *bad* if $|T_v| \notin (1\pm\delta)\frac{1}{2}|S_v|$ and *good* otherwise. If v is good, let T'_v denote an arbitrary subset of T_v of cardinality $(1-\delta)\frac{1}{2}|S_v|$. Similarly, let S'_v denote an arbitrary subset of S_v of cardinality $(1-\delta)|S_v|$, so that $|T'_v| = \frac{1}{2}|S'_v|$. Next, choose an arbitrary 2-matching from T'_v to S'_v. Note that all matched edges are of the form (b, r), where $r = b - \lambda(1+2/\delta)\cdot 1_T$ for some $\lambda \in (0, 2C]$. This completes the definition of the 2-matching M_T for a fixed set T.

We now argue that there cannot be too many bad classes in a fixed set T. Note that there are $\Omega(m^{2m})$ equivalence classes (since they have constant size by construction). For a vertex v denote the event that v's equivalence class is bad by $\mathcal{E}_v$. Then, conditional on the value of Q_T, these events are independent for non-equivalent v's. Hence, by Chernoff bounds the probability that the number of bad classes exceeds its expectation by more than a factor of 4 is at most $e^{-\Omega(m^{2m})}$. We use the collection $\mathcal{F}$ constructed in the proof of Lemma 48, and a union bound over $2^{O(m)}$ sets T shows that there will be no more than a δ fraction of bad classes in any of sets T with high probability.

We will also need a bound on the maximum degree of vertices in X and Y. First note that the definition of the set of levels L_s and the random variable Q_T amounts to flipping the role of the sets R_k and B_k independently with probability $1/2$. Thus, for a fixed T, every vertex except for those in $W \cup W'$, of which there is only an $O(\delta)$ fraction, takes part in the matching with probability $1/2$. Thus, the expected degree of a fixed vertex $v \in Y$ is at most $|\mathcal{F}|/2$, where $\mathcal{F}$ is the collection of almost orthogonal vectors that we use. Since Q_T are independent for each T, Chernoff bounds imply that the degree of any vertex v in Y in the graph that we construct is at most $(1+\delta)|\mathcal{F}|/2$ with probability at least $1 - e^{-\Omega(|\mathcal{F}|)} = 1 - e^{-\Omega(2^{\Omega(m)})}$. In particular, a union bound over all $v \in Y$, of which there are m^{2m}, shows that the degree cannot be larger than $(1+\delta)|\mathcal{F}|/2$ for any v with high probability. Finally, we also note that the average degree is at least $(1-O(\delta))|\mathcal{F}|/2$ by construction. A similar argument shows that the maximum degree of a vertex in X does not exceed $(1+\delta)|\mathcal{F}|$ with high probability, and the average degree is at least $(1-O(\delta))|\mathcal{F}|$.

Essentially the same argument as in Lemma 48 together with the fact that for good sets T we have a 2-matching of at least $(1-O(\delta))|R|$ nodes by the argument above shows that the size of the matching is at least $(\frac{1}{2} - O(\delta))|X|$.

Now let T_1, T_2 be two sets in $[m]$ of size $p = (\delta/(8C))m$ such that $|T_1 \cap T_2| \leq (5/2)(\delta/(8C))^2 m$. We show that no edge of M_{T_1} is induced by M_{T_2}. Let b be matched to r by T_1, i.e. $b - r = j(2/\delta+1)1_{T_1}$ for some $j \in (0, 2C]$. If the edge (b, r) is induced by M_{T_2}, then one of b, r is colored blue and the other is colored red in the coloring induced by T_2. In particular, b and r are separated by a white strip. Thus,

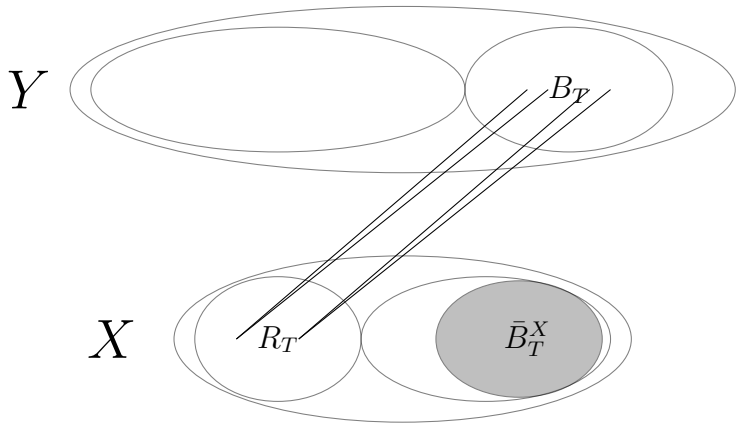
$$\left|\sum_{i\in T_2} b_i - \sum_{i\in T_2} r_i\right| \geq (\delta/(8C))m. \tag{3.40}$$

On the other hand,

$$\begin{aligned}
&\left|\sum_{i\in T_2} b_i - \sum_{i\in T_2} r_i\right| = \left|\sum_{i\in T_2}(b-r)_i\right| \\
&= \left|\sum_{i\in T_2}(j(2/\delta+1)1_{T_1})_i\right| \\
&\leq 2C(2/\delta+1)|T_1\cap T_2| < C(2/\delta+1)(5/2)(\delta/(8C))^2 \\
&\leq (5/6)(1+\delta/12)(\delta/8C)m,
\end{aligned} \tag{3.41}$$

a contradiction with (3.40) for any $\delta \leq 1/2$. This completes the proof of (1).

Figure 3.5: A 2-matching M_T



It remains to show (2). Consider a fixed matching M_T. Let $R_T = X\cap M_T \subseteq R^X, B_T = Y\cap M_T \subseteq B^Y$, where we use the notation B^X, B^Y, R^X, R^Y to denote the set of blue and red points in X and Y respectively. For a vertex $u \in X\cup Y$, denote by $\Gamma(u)$ its neighbors in G. Let

$$\bar{B}_T = \bigcup_{k} \bigcup_{s=kw+C(2+2/\delta)p}^{kw+C(4/\delta)p-1} L_s.$$

Note that $\bar{B}_T \subset B_T$ can be viewed as the 'interior' of B_T. We write $\bar{B}_T^X$ and $\bar{B}_T^Y$ to denote the projection of $\bar{B}$ onto X and Y respectively.

We first show that for all $x \in \bar{B}_T^X$ one has $\Gamma(x) \subseteq B_T \subseteq Y$. Since $\bar{B}_T^X$ is not matched by T, it suffices to consider edges of $M_{T'}, T' \neq T$. But any such edge has the form (x, y), where $x = y \pm \lambda(2/\delta+1)\cdot 1_{T'}$, so by the argument above one has

$$\left|\sum_{i\in T_2} x_i - \sum_{i\in T_2} y_i\right| < (\delta/(8C))m = p, \tag{3.42}$$

so $y \in B_T^Y$.

Thus, for each $x \in \bar{B}_T^X$ one has $\Gamma(x) \subseteq B_T^Y$. We can now exhibit the required fractional matching. Include edge $(x, y), r \in B_T^X, y \in \bar{B}_T^Y$ with weight $\frac{1}{(1+O(\delta))|\mathcal{F}|}$, and include all edges of the 2-matching M_T with weight 1/2. Since the maximum degree of a node in B_T is at most $(1+\delta)|\mathcal{F}|/2$, and the maximum degree of a node in $\bar{B}_T$ is at most $(1+O(\delta))|\mathcal{F}|$, this assignment yields a feasible fractional matching. Recall that by construction, the average degree in X is at least $(1-O(\delta))|\mathcal{F}|/2$, hence the size of the fractional matching is at least $(1-O(\delta))|X|$.

By the integrality of the matching polytope, the fractional matching can be rounded to produce an integral matching of size at least $(1-O(\delta))|X|$, as required. Note that since we proved that for each $x \in \bar{B}_T^X$ one has $\Gamma(y) \subseteq B_T^Y$, the fractional matching that we constructed avoids $Y \setminus B_T = Y \setminus (Y \cap M_T)$, and hence so does the integral matching. This completes the proof of (2).

Finally, we note that the number of edges in the graph is given by $n^{1+\Omega_\delta(1/\log\log n)}$, as before. ■

We note that the same techniques can be used to prove the following more general

Lemma 52 *For any fixed constants $\epsilon, \gamma > 0$ and an arbitrarily small constant $\delta > 0$ there exists a family of bipartite Ruzsa-Szemerédi graphs $G = (X, Y, E)$ with $|X| = n$, $|Y| = n/\epsilon$ such that*

1. *the edge set E is a union of $n^{\Omega_{\epsilon,\delta,\gamma}(1/\log\log n)}$ induced $\frac{1-\gamma}{\epsilon\gamma}$-matchings $M_1, \ldots, M_k$ of size at least $(\gamma-\delta)|X|$.*
2. *for every $j \in [1:k]$ the graph G contains a matching M_j^* of size at least $(1-O(\delta))|X|$ that avoids $Y \setminus (M_j \cap Y)$.*

3.7 Lower bounds on communication and one-pass streaming complexity

We show here that lower bounds on the size of Ruzsa-Szemerédi graphs yield lower bounds on the (randomized) communication complexity, and hence for one-pass streaming complexity.

In the edge model, we show that $CC\left(\frac{2(1-\epsilon)}{2-\epsilon} - \delta, (2-\epsilon)n\right) = \Omega(U_I(\epsilon, n))$ for all $\epsilon, \delta > 0$. In particular, combined with the constructions of $(1/2+\delta_0)$-Ruzsa-Szemerédi graphs for any constant $\delta_0 > 0$ (Lemma 48) this proves that $CC(\epsilon, n) = n^{1+\Omega(1/\log\log n)}$ for $\epsilon < 1/3$. Thus our $O(n)$ upper bound on $CC(\frac{1}{3}, n)$ in section 3.3 is optimal in the sense that any better approximation requires super-linear communication. As a corollary, we also get that super-linear space is necessary to achieve better than 2/3-approximation in the one-pass streaming model.

In the vertex model, using the construction of Ruzsa-Szemerédi graphs from Lemma 49, we show that $CC_v(\epsilon, n) = n^{1+\Omega(1/\log\log n)}$ for all $\epsilon < 1/4$. This proves optimality of our construction in section 3.4, and also shows that super-linear space is necessary to achieve better than 3/4-approximation

in the one-pass streaming model even in the vertex arrival setting.

We note that our lower bounds for both the edge and vertex arrival case apply to randomized algorithms. The proofs of these results appear in the full version.

3.7.1 Edge arrivals

Lemma 53 *For any $\epsilon > 0$ and $\delta > 0$, $CC\left(\frac{2(1-\epsilon)}{2-\epsilon} - \delta, (2-\epsilon)n\right) = \Omega(U_I(\epsilon, n))$.*

Proof: For any $\delta > 0$, we will construct a distribution over bipartite graphs with $(2-\epsilon)n$ vertices on each side such that each graph in the distribution contains a matching of size at least $(2-\epsilon)n - \delta n$. On the other hand, we will define a partition of the edge set E of the graph into $E = E_1 \cup E_2$ and show that any for deterministic communication protocol using message size $s = o(U_I(\epsilon, n))$, the expected size of the matching computed is bounded by $2(1-\epsilon)n + o(n)$. Using Yao's minmax principle, we get the desired performance bound for any protocol with $o(U_I(\epsilon, n))$ communication.

Let $G = (P, Q, E)$ be an ϵ-RS graph with n vertices on each side and $U_I(\epsilon, n)$ edges. By definition, E can be partitioned into k induced matchings $M_1, ..., M_k$, where $|M_i| = \epsilon n$ for $1 \leq i \leq k$, and $k = U_I(\epsilon, n)/(\epsilon n)$. We generate a random bipartite graph $G' = (P_1 \cup P_2, Q_1 \cup Q_2, E_1 \cup E_2)$ with $(2-\epsilon)n$ vertices on each side, as follows:

1. We set $P_1 = P$ and $Q_1 = Q$. Also, let P_2 and Q_2 be a set of $(1-\epsilon)n$ vertices each that are disjoint from P and Q.

2. For each M_i, $i = 1, ..., k$, let M_i' be a uniformly at random chosen subset of M_i of size $(1-\delta)n$. We set $E_1 = \cup_{i=1}^{k} M_i'$.

3. Choose a uniformly random $r \in [1:k]$. Let M_1^* be an arbitrary perfect matching between P_2 and $Q \setminus Q_1(M_r)$, and let M_2^* be an arbitrary perfect matching between Q_2 and $P \setminus P_1(M_r)$. We set $E_2 = M_1^* \cup M_2^*$.

The instance G' is partitioned between Alice and Bob as follows: Alice is given all edges in $G_1(P_1, Q_1, E_1)$ (first phase), and Bob is given all edges in $G_2(P_2, Q_2, E_2)$ (second phase). Clearly, any optimal matching in G' has size at least $(2-\epsilon)n - \delta n$; consider, for instance, the matching $M_r' \cup M_1^* \cup M_2^*$.

We now show that for any deterministic communication protocol using communication at most $s = o(U_I(\epsilon, n))$, with probability at least $(1 - o(1))$, number of edges in M_r' retained by the algorithm at the end of the first phase is $o(n)$. Assuming this claim, we get that with probability at least $(1 - o(1))$, the size of the matching output by Bob is bounded by $2(1-\epsilon)n + o(n)$. Hence the expected size of the matching output by Bob is bounded by $2(1-\epsilon)n + o(n)$. We now establish the preceding claim.

We start by observing that the number of distinct first phase graphs is at least (assume $\delta < \epsilon/2$)

$$\binom{\epsilon n}{\delta n}^k = \binom{\epsilon n}{\delta n}^{\frac{U_I(\epsilon,n)}{\epsilon n}} = 2^{\gamma U_I(\epsilon,n)},$$

for some positive γ bounded away from 0. Let $\mathcal{G}$ denote the set of all possible first phase graphs, and let $\phi : \mathcal{G} \to \{0,1\}^s$ be the mapping used by Alice to map graphs in $\mathcal{G}$ to a message of size $s = o(U_I(\epsilon, n))$. For any graph $H \in \mathcal{G}$, let $\Gamma(H) = \{H' \mid \phi(H') = \phi(H)\}$. Then note that for any graph $H \in \mathcal{G}$, Bob can output an edge e in the solution iff e occurs in every graph $H' \in \Gamma(H)$. For any subset F of $\mathcal{G}$, let G_F denote the unique graph obtained by intersection of all graphs in F (i.e. the graph G_F contains an edge e iff e is present in every graph in the family F).

Claim 54 *For any $0 < \epsilon' < \frac{\epsilon}{2}$ and any subset F of $\mathcal{G}$, let $I \subseteq \{1, 2, ..., k\}$ be the set of indices such that G_F contains at least $\epsilon' n$ edges from M_i for each $i \in I$. Then if $|F| \geq 2^{(\gamma - o(1))U_I(\epsilon,n)}$, $|I| = o(k)$.*

Proof: Let $|I| = k_1$. Then the number of graphs that can be in F is bounded by

$$\binom{(\epsilon - \epsilon')n}{\delta n}^{k_1} \binom{\epsilon n}{\delta n}^{k-k_1} = \left(2^{-\Omega(\epsilon' n)} \binom{\epsilon n}{\delta n}\right)^{k_1} \binom{\epsilon n}{\delta n}^{k-k_1} = 2^{-\Omega(k_1(\epsilon' n))} \binom{\epsilon n}{\delta n}^{k}.$$

It then follows that if $k_1 = \Omega(k)$, we have $|F| \leq 2^{(\gamma - \Omega(1))U_I(\epsilon,n)}$, contradicting our assumption on the size of F. ■

To conclude the proof, we note that a simple counting argument shows that for a uniformly at random chosen graph $H \in \mathcal{G}$, with probability at least $1 - o(1)$, we have $|\Gamma(H)| \geq 2^{(\gamma - o(1))U_I(\epsilon,n)}$. Conditioned on this event, it follows from Claim 54 that for a randomly chosen index $r \in [1..k]$, with probability at least $1 - o(1)$, the graph $G_{\Gamma(H)}$ contains at most $\epsilon' n$ edges from M_r. ■

In particular, we get

Corollary 55 *For any $\delta > 0$, $CC(2/3 + \delta, n) = n^{1+\Omega_\delta(1/\log\log n)}$.*

Proof: Follows by putting together Lemma 48 and Lemma 53. ■

Lower bounds on communication complexity translate directly into bounds on one-pass streaming complexity:

Corollary 56 *For any constant $\delta > 0$ any (possibly randomized) one-pass streaming algorithm that achieves approximation factor $\frac{2(1-\epsilon)}{2-\epsilon} + \delta$ must use $\Omega(U_I(\epsilon, n))$ space. In particular, any one-pass streaming algorithm that achieves approximation factor $2/3 + \delta$ must use $n^{1+\Omega_\delta(1/\log\log n)}$ space.*

Proof: Follows by Lemma 48 and Lemma 53. ■

3.7.2 Vertex arrivals

We now prove a lower bound on the communication complexity in the vertex arrival model using the construction of lop-sided Ruzsa-Szemerédi graphs from Lemma 49. The bound implies that our upper bound from section 3.4 is tight. Moreover, the bound yields the first lower bound on the streaming complexity in the vertex arrival model.

Lemma 57 *For any constant $\delta > 0$, $CC_v^1(3/4 + \delta, n) = n^{1+\Omega_\delta(1/\log\log n)}$.*

Proof: For sufficiently small $\delta > 0$, we will construct a distribution over bipartite graphs with $(2+\delta)n$ vertices on each side such that each graph in the distribution contains a matching of size at least $(2 - O(\delta))n$. On the other hand, we will show that for any deterministic protocol using space $s = n^{1+o(1/\log\log n)}$, the expected size of the matching computed is bounded by $(3/2+O(\delta))n+o(n)$. Using Yao's minmax principle we get the desired performance bound for any $n^{1+o(1/\log\log n)}$-space randomized protocol.

Let $G = (P, Q, E)$ be an $(1/2-\delta)$-RS graph with $|P| = n, |Q| = 2n$ and $n^{1+\Omega(1/\log\log n)}$ edges, as guaranteed by Lemma 51. By definition, E can be partitioned into k induced 2-matchings $M_1, ..., M_k$, where $|M_i| \geq (1/2 - \delta')n$ for $1 \leq i \leq k$, and $k = n^{\Omega(1/\log\log n)}$ and some $\delta' = O(\delta)$. We generate a random bipartite graph $G' = (P_1 \cup P_2, Q, E_1 \cup E_2)$ with $(2 + \delta')n$ vertices on each side, as follows:

1. We set $P_1 = P$ and let P_2 be a set of $(1 + \delta')n$ vertices that are disjoint from P.
2. For each M_i, $i = 1, ..., k$, let M'_i be a uniformly at random chosen subset of M_i of size $(1/2 - 2\delta')n$. We set $E_1 = \cup_{i=1}^{k} M'_i$.
3. Choose a uniformly random $r \in [1 : k]$. Let M^* be an arbitrary perfect matching between P_2 and $Q \setminus Q(M_r)$. We set $E_2 = M^*$.

Let Alice hold the graph $G_A(P_1, Q_1, E_1)$ and let Bob hold the graph $G_2 = (P_2, Q, E_2)$. By Lemma 49, there exists a matching M^*_r that matches at least a $(1 - \delta')$ fraction of X and avoids $Q \setminus Q(M_r)$. Thus, any optimal matching in $G_A \cup G_B$ has size at least $(2 - O(\delta))n$; consider, for instance, the matching $M^*_r \cup M^*$.

However, no deterministic space protocol can output more than a $\delta'' = O(\delta')$ fraction of the edges in M'_r if it uses $n^{1+o_{\delta''}(1/\log\log n)}$ space by the same argument as in 53. Hence, the size of the matching output by the protocol is bounded above by $(1/2 + O(\delta))|P_1| + |P_2| = (3/2 + O(\delta))n$. ■

We immediately get

Corollary 58 *For any constant $\delta > 0$ any (possibly randomized) one-pass streaming algorithm that achieves approximation factor $3/4 + \delta$ must use $n^{1+\Omega_\delta(1/\log\log n)}$ space.*

3.8 Matching covers versus Ruzsa-Szemerédi graphs

In this section we prove that the size of the smallest possible matching cover is essentially the same as the number of edges in the largest Ruzsa-Szemerédi graph with appropriate parameters.

We are now ready to state the two theorems that use induced matchings to bound the size of matching covers. The lower bound is easy, and is proved first. The upper bound is more intricate, and is presented in section 3.8.1.

Theorem 59 [Lower bound] *For any $\delta > 0$, $L_C(\epsilon, n) \geq U_I((1 + \delta)\epsilon, n) \cdot \left(\frac{\delta}{1+\delta}\right)$.*

Proof of Theorem 59: Let $c = 1 + \delta$. By definition, there exists an undirected bipartite graph $G = (P, Q, E)$ with $|E| = U_I(\epsilon c, n)$, $|P| = |Q| = n$, and an induced partition $\mathcal{F}$ of G such that every

set in the partition is of size at least ϵcn. Consider the smallest ϵ-matching-cover H of G, and any set $F \in \mathcal{F}$. Recall that by the definition of an induced matching, the edges in F are the only edges between $P(F)$ and $Q(F)$. Since F is a matching between $P(F)$ and $Q(F)$, and the size of F is at least ϵcn, the intersection of H and F must be of size at least $|F| - \epsilon n$, which is at least $|F| \cdot \left(\frac{c-1}{c}\right)$. Summing over all sets F in the partition $\mathcal{F}$, we get that $|H| \geq |E| \cdot \left(\frac{c-1}{c}\right)$, which proves the theorem. ■

In particular, choosing $\delta = 1$, we get $L_C(\epsilon, n) \geq U_I(2\epsilon, n)/2$. The upper bound is more complicated; we first state a simplified version (Theorem 60), and then the full version (Theorem 61). The simple version is a corollary of the full version; the full version is proved in section 3.8.1.

Theorem 60 [Simplified upper bound] *Assume $0 < \epsilon < 2/3, 0 < \delta < 1$, and $\epsilon n \geq 3$. Then, $L_C(n, \epsilon) \leq U_I((1-\delta)\epsilon, n) \cdot O\left(\frac{\log(1/\epsilon)}{\delta(1-\delta)}\right)$.*

Theorem 61 [Upper bound] *Assume $\epsilon n \geq 3$, and $0 < \delta < 1$. Then,*

$$L_C(n, \epsilon) \leq U_I((1-\delta)\epsilon, n) \cdot \left(\frac{8\epsilon n}{\epsilon n - 1}\right) \cdot \left(1 + \log(1/\epsilon) + \frac{\log(\epsilon n)}{8\epsilon n}\right) \cdot \left(\frac{1}{\delta(1-\delta)}\right).$$

We state the full expression in the above theorem as opposed to using asymptotic notation since the constants are simple, and it is conceivable that one may choose to apply it in regimes where ϵ is arbitrarily close to 1. Choosing $\delta = 1/2$ in Theorem 60, we get the interesting special case, $L_C(n, \epsilon) = O(U_I(\epsilon/2, n) \log(1/\epsilon))$.

3.8.1 Proof of the Upper Bound

We will now prove Theorem 61. Assume we are given an arbitrary undirected bipartite graph $G = (P, Q, E)$ with $|P| = |Q| = n$. Assume that ϵn is an integer. Also assume that ϵn is at least 3 (of course the most interesting case is when $\epsilon > 0$ is some constant). Before proceeding, we need another definition:

Definition 62 *A pair (A, B), where $A \subseteq P$ and $B \subseteq Q$, is said to be "critical" if $|A| = |B| = M_E(A, B) = \epsilon n$, i.e. A, B are both of size ϵn and there is a perfect matching between them. Let $\mathcal{C}$ denote the set of all critical pairs in G.*

We will now consider a primal-dual pair of Linear Programs. By strong duality, the optimum objective value for both LPs is the same; denote this value as Z^*. We label the constraints in the primal with the corresponding variable in the dual, and vice versa, for clarity.

$$
\begin{array}{lll}
\text{PRIMAL:} & Z^* = \text{minimize} \sum_{e \in E} x_e & \\
\quad \text{Subject to:} & & \\
\forall (A,B) \in \mathcal{C}: & \sum_{e \in E \cap (A \times B)} x_e \geq 1 & [\lambda_{A,B}] \\
& x \geq 0 &
\end{array}
$$

$$
\begin{array}{lll}
\text{DUAL:} & Z^* = \text{maximize} \sum_{(A,B) \in \mathcal{C}} \lambda_{A,B} & \\
\quad \text{Subject to:} & & \\
\forall (e) \in E: & \sum_{(A,B) \in \mathcal{C}: e \in E \cap (A \times B)} \lambda_{(A,B)} \leq 1 & [x_e] \\
& \lambda \geq 0 &
\end{array}
$$

We will relate the size of an ϵ-matching-cover of G to the primal and the size of an ϵ-induced partition of G to the dual. In particular, in the next two subsections, we will prove the following two lemmas:

Lemma 63 *The graph G has an ϵ-matching-cover of size at most*

$$\left(\frac{\epsilon n}{\epsilon n - 1}\right) \cdot (2\epsilon n(1 + \log(1/\epsilon)) + \log(\epsilon n)) \cdot Z^*.$$

Lemma 64 *There exists a graph $G' = (P, Q, E')$ with $E' \subseteq E$ such that $|E'| \geq Z^* \delta(1-\delta)\epsilon n/4$ edges, and G' has a $(1-\delta)\epsilon$-induced partition. Hence, $U_I(n, (1-\delta)\epsilon) \geq Z^* \delta(1-\delta)\epsilon n/4$.*

Theorem 61 is immediate from these two lemmas.

Proof of Lemma 63

A set of edges $F \subseteq E$ is said to satisfy a pair (A, B) if $|F \cap (A \times B)| > 0$. We will further break down the proof of Lemma 63 in two parts.

Lemma 65 *If F satisfies all critical pairs, then F is an ϵ-matching-cover.*

Proof: The proof is by contradiction. Suppose F satisfies all critical pairs, but there exists a pair (A, B) such that $A \subseteq P$, $B \subseteq Q$, and $M_F(A, B) < M_E(A, B) - \epsilon n$. Consider an arbitrary maximum matching in the graph $(A, B, E \cap (A \times B))$, say H. Discard all vertices from A and B that are not incident on an edge in H, to obtain $A' \subseteq A$, $B' \subseteq B$. It is still true that $M_F(A', B') < M_E(A', B') - \epsilon n$, but now we also know that $M_E(A', B') = |H| = |A'| = |B'|$. Consider the graph $G' = (A', B', F)$. By Hall's theorem, there exists a set $A'' \subseteq A'$ and another set $B'' \subseteq B'$ such that (a) $|A''| > |B''| + \epsilon n$, and (b) $|F \cap (A'' \times (B' \setminus B''))| = 0$. Since H is perfect matching in the graph (A', B', E), there must exist at least ϵn edges of H that go from A'' to $B' \setminus B''$; let H' denote an arbitrary set of ϵn edges of H that go from A'' to $B' \setminus B''$. Let C denote the endpoints of these edges

in P and D denote the endpoints of these edges in Q. Then, $|C| = |D| = \epsilon n$ and there is a perfect matching between C and D in E, i.e., the pair (C, D) is critical. But there is no edge between C and D in F (by construction), and hence F does not satisfy all critical pairs, which contradicts our assumption. ■

Lemma 66 *There exists a set F of size at most*

$$\left(\frac{\epsilon n}{\epsilon n - 1}\right) \cdot (2\epsilon n(1 + \log(1/\epsilon)) + \log(\epsilon n)) \cdot Z^*$$

that satisfies all critical pairs.

Proof: First note that the number of critical pairs is at most $\binom{n}{\epsilon n}^2 < \left(\frac{en}{\epsilon n}\right)^{2\epsilon n} = e^{2\epsilon n(1+\log(1/\epsilon))}$.

We will now define a simple randomized rounding procedure for the solution x of the primal LP. For convenience, let γ denote the quantity $(2\epsilon n(1 + \log(1/\epsilon)) + \log(\epsilon n))$. For each edge e, let $\tilde{x}_e$ denote a Bernoulli random variable which takes the value 1 with probability $p_e = \min\{1, \gamma x_e\}$, and let all $\tilde{x}_e$'s be independent. Let F denote the set of edges e for which $\tilde{x}_e = 1$.

We will now define two bad events: Let ξ_1 denote the event that $|F| > \gamma Z^* \left(\frac{\epsilon n}{\epsilon n - 1}\right)$. Let ξ_2 denote the event that F does not satisfy all critical sets.

By construction, $\mathbf{E}[|F|] = \mathbf{E}[\sum_e \tilde{x}_e] \leq \gamma \sum_e x_e = \gamma Z^*$. Hence, by Markov's inequality, $\mathbf{Pr}[\xi_1] < \frac{\epsilon n - 1}{\epsilon n} = 1 - 1/(\epsilon n)$.

Fix an arbitrary critical set (A, B). If there exists an edge $e \in E \cap (A \times B)$ such that $p_e = 1$ then (A, B) is deterministically satisfied by F. Else, it must be that $p_e = \gamma x_e$ for every edge $e \in E \cap (A \times B)$, and the probability that F does not satisfy (A, B) is at most

$$\begin{aligned}
&\prod_{e \in E \cap (A \times B)} (1 - \gamma x_e) \\
&\leq e^{-\gamma \sum_{e \in E \cap (A \times B)} x_e} \\
&\leq e^{-\gamma} \qquad \text{[From feasibility of the fractional solution]}.
\end{aligned}$$

Using the union bound over all critical pairs, we get $\mathbf{Pr}[\xi_2] < e^{-\log(\epsilon n)} = 1/(\epsilon n)$. Using the union bound over the two bad events, we get $\mathbf{Pr}[\xi_1 \cup \xi_2] < 1$. Hence, (using the probabilistic method), there must exist a set of edges F that satisfies all critical pairs and has size at most $\left(\frac{\epsilon n}{\epsilon n - 1}\right) \cdot (2\epsilon n(1 + \log(1/\epsilon)) + \log(\epsilon n)) \cdot Z^*$. ■
This concludes the proof of Lemma 63.

Proof of Lemma 64

This proof is also via randomized rounding, this time applied to the optimum solution of the dual LP. For every relevant pair (A, B), choose $\tilde{\lambda}_{A,B}$ to be one with probability $\delta\lambda_{A,B}/2$ and 0 otherwise; further choose the values of different $\tilde{\lambda}_{A,B}$'s independently. If $\tilde{\lambda}_{A,B} = 1$ then we say that the pair (A, B) has been selected. Initialize H to be E; we will remove edges from H till the graph (P, Q, H) has an ϵ-induced partition.

Step 1: Getting an induced partition. First, fix an arbitrary perfect matching (in E) between each selected pair, and (a) remove all edges from H that do not belong to any of these perfect matchings. Then, (b) remove all edges that belong to more than one of the graphs induced by the selected pairs. Let the new set of edges be called H_1.

Step 2: Pruning small induced sets. At this point, the collection of sets of edges $\{(A\times B)\cap H_1 : \tilde{\lambda}_{A,B} = 1\}$ forms an induced partition of the graph (P, Q, H_1). The only problem is that some of the sets in this partition may be too small. We will count a selected pair (A, B) as "good" if it induces at least $(1-\delta)\epsilon n$ edges in H_1, and "bad" otherwise. Remove all edges from H_1 that are induced by a bad selected pair to obtain the set H_2. The set (P, Q, H_2) now has a $((1-\delta)\epsilon)$-induced partition. Let k denote the number of good selected pairs; then $|H_2|$ (and hence $U_I(n, (1-\delta)\epsilon)$) is at least $k(1-\delta)\epsilon n$.

We will now show that $\mathbf{Pr}[k > \delta Z^*/4] > 0$. Consider a relevant pair (A, B) with $\lambda_{A,B} > 0$. Now, $\mathbf{Pr}[\tilde{\lambda}_{A,B} = 1] = \delta\lambda_{A,B}/2$. Consider the perfect matching F chosen between this pair (arbitrarily) in step 1 and consider any edge e in this matching. This edge will not be pruned away in step 1(a). By the feasibility constraint in the dual,

$$\sum_{(A',B')\in\mathcal{C}:(A,B)\neq(A',B'),e\in E\cap(A'\times B')} \lambda_{A',B'} < 1.$$

Hence, the probability that this edge will belong to a selected pair other than (A, B) is less than $\delta/2$. Thus, the expected number of edges in $H_1 \cap (A \times B)$ is more than $(1-\delta/2)\epsilon n$. The maximum number of edges in $H_1 \cap (A \times B)$ is ϵn. Applying Markov's inequality to the random variable $\epsilon n - |H_1 \cap (A \times B)|$, we get:

$$\mathbf{Pr}[|H_1 \cap (A \times B)| \geq (1-\delta)\epsilon n \mid \tilde{\lambda}_{A,B} = 1] > 1/2.$$

Multiplying with the probability that $\tilde{\lambda}_{A,B} = 1$, we obtain:

$$\mathbf{Pr}[\text{A relevant pair } (A, B) \text{ is both selected and good}] > \delta\lambda_{A,B}/4.$$

Summing over all relevant pairs (A, B), we get $\mathbf{E}[k] > \delta Z^*/4$, and hence (using the probabilistic method again), there must exist a set of choices for $\tilde{\lambda}_{A,B}$ which make $k > \delta Z^*/4$. For this choice, we know that H_2 (and hence $U_I(n, (1-\delta)\epsilon)$) is at least $Z^*\delta(1-\delta)\epsilon n/4$.

This concludes the proof of Lemma 64.

Finally, we note that an upper bound on the size of ϵ-covers directly yields an upper bound on the communication complexity of achieving an *additive* ϵn error approximation to bipartite matching, denoted by $CC_+(\epsilon, n)$.

Lemma 67 $CC_+(\epsilon, n) \leq L_C(\epsilon, n)$.

Proof: Let $G_1 = (P_1, Q_1, E_1)$ denote the bipartite graph with $|P| = |Q| = n$ that Alice holds and let $G_2 = (P_2, Q_2, E_2)$ be the graph that Bob holds. Let G_1' be a ϵ-matching cover of G_1. Consider

an empty cut $(A_1 \cup B_1, A_2 \cup B_2)$ corresponding to a maximum matching M' in $(G'_1 \cup G_2)$, i.e. such that $|M'| = |B_1| + |A_2|$. Let M^* denote a maximum matching in $(A_1 \times B_2) \cap E_1$. Since G'_1 is an ϵ-matching cover, we have that $|M^*| < \epsilon n$.

Thus, since the maximum matching M in $G_1 \cup G_2$ is bounded by $|B_1| + |A_2| + |M^*|$ we have

$$|M| - |M'| \leq (|B_1| + |A_2| + |M^*|) - (|B_1| + |A_2|) \leq \epsilon n.$$

■

Chapter 4

$1 - 1/e$ lower bound and multipass algorithms

In this chapter we build upon our techniques introduced in Chapter 3 to obtain the following results. First, we prove an optimal impossibility result for approximating matchings in a single pass in the streaming model. In particular, we show that no algorithm that uses quasilinear space can achieve a $1 - 1/e + \delta$ approximation to the size of the matching in a single pass for any constant $\delta > 0$. We then devise a simple multipass algorithm for approximating maximum matchings using linear space that improves upon the state of the art. Finally, we show how our techniques apply to the Gap-Existence problem, which we define below.

The problem of approximating maximum matchings in bipartite graphs has received significant attention recently, and very efficient small-space solutions are known when multiple passes are allowed[28, 68, 26, 3, 4, 62]. The best known algorithm due to Ahn and Guha [3] achieves a $1 - O(\sqrt{\log\log k/k})$ in k passes for the weighted as well as the unweighted version of the problem using $\tilde{O}(kn)$ space.

Single pass algorithms. All algorithms mentioned above require at least two passes to achieve a nontrivial approximation. The problem of approximating matchings in a single pass has recently received significant attention[35, 62]. Two natural variants of this problem have been considered in the literature: (1) the edge arrival setting, where edges arrive in the stream and (2) the vertex arrival setting, when vertices on one side of the graph arrive in the stream together with all their incident edges. The latter setting has also been studied extensively in the context of *online algorithms*, where each arriving vertex has to either be matched irrevocably or discarded upon arrival.

In a single pass, the best known approximation in the edge arrival setting is still $1/2$, achieved by simply keeping a maximal matching (this was recently improved to $1/2 + \epsilon$ for a constant $\epsilon > 0$ under the additional assumption of random edge arrivals[62]). We showed in Chapter 3 that no $\tilde{O}(n)$ space algorithm can achieve a better than $2/3$ approximation in this setting.

In the vertex arrival setting, the best known algorithms achieve an approximation of $1 - 1/e$.

The assumption of vertex arrivals allows one to leverage results from online algorithms [56, 67, 50] . In the online model vertices on one side of the graph are known, and vertices on the other side arrive in an adversarial order. The algorithm has to either match a vertex irrevocably or discard upon arrival. The celebrated algorithm of Karp-Vazirani-Vazirani achieves a $1-1/e$ approximation for the online problem by crucially using randomization (additionally, this algorithm only uses $\tilde{O}(n)$ space). A *deterministic* single pass $\tilde{O}(n)$ space $1-1/e$ approximation in the vertex arrival setting was given in Chapter 3 (such a deterministic solution is provably impossible in the online setting). We also showed by analyzing a communication complexity problem that no single-pass streaming algorithm that uses $\tilde{O}(n)$ space can obtain a better than $3/4$ approximation in the vertex arrival setting. The protocol provided a protocol for this communication problem that matches the $3/4$ approximation ratio, suggesting that new techniques would be needed to prove a stronger impossibility result.

Lop-sided graphs. The techniques for matching problems outlined above yield efficient solutions that use $\tilde{O}(|P| + |Q|)$ space, where $|P|$ and $|Q|$ are the sizes of the sets in the bipartition. While this is a reasonable space bound to target, this can be prohibitively expensive for lop-sided graphs that arise, for example, in applications to ad allocations. Here the P side of the graph corresponds to the set of *advertisers*, and the Q side to the set of *impressions* [21]. An important constraint is that the set of impressions Q may be so large that it is not feasible to represent it explicitly, ruling out algorithms that take $O(|P| + |Q|)$ space.

Data model for lop-sided graphs. Since the set Q cannot be represented explicitly, it is important to fix the model of access to Q. Here we assume the following scenario. Vertices in P arrive in the stream in an adversarial order, together with a representation of their edges. We make no assumptions on the way the edges are represented. For example, some edges could be stored explicitly, while others may be represented implicitly. We assume access to the following two functions:

1. LIST-NEIGHBORS(u, S) which, given a set of vertices $S \subseteq Q$ and a vertex $u \in P$, lists the neighbors of u in S;
2. NEW-NEIGHBOR(u, S) which, given a set of vertices $S \subseteq Q$ and a vertex $u \in P$ outputs a neighbor of u outside of the set S.

In this chapter, we improve upon the best known bounds for both the single pass and multi-pass settings. In the single pass setting, we prove an optimal impossibility result for vertex arrivals, which also yields the best known impossibility result in the edge arrival model. For the multipass setting, we give a simple algorithm that improves upon the approximation obtained by Ahn and Guha in the vertex arrival setting, as well as yields an efficient solution to the Gap-Existence problem considered by Charles et al[21].

Lower bounds. In this paper we build upon the communication complexity approach taken in Chapter 3 to obtain lower bounds via what can be viewed as multi-party communication complexity. Our main result is an optimal bound on the best approximation ratio that a single-pass $\tilde{O}(n)$ space streaming algorithm can achieve in the vertex arrival setting:

Theorem 68 *No (possibly randomized) one-pass streaming algorithm that outputs a valid matching with probability at least 3/4 can obtain a better than $1 - 1/e + \delta$-approximation to the maximum matching, for any constant $\delta > 0$, unless it uses at least $n^{1+\Omega(1/\log\log n)}$ space, even in the vertex arrival model.*

We note that this bound is matched by the randomized KVV algorithm[56] for the online problem and the deterministic $\tilde{O}(n)$ space algorithm of Chapter 3. One striking consequence of our bound is that no single-pass streaming algorithm can improve upon the more constrained *online* algorithm of KVV, which has to make irrevocable decisions, unless is uses significantly more than $\tilde{O}(n)$ space. Our bound also improves upon the best known bound of 2/3 for small space one-pass streaming algorithms in the *edge arrival model*.

We showed in Chapter 3 via an analysis of the natural two-party communication problem that no one-pass streaming algorithm that uses $\tilde{O}(n)$ space can achieve approximation better than 2/3 in the edge arrival setting and 3/4 in the vertex arrival setting. Furthermore, we also gave a communication protocol that proves the optimality of both bounds for the communication problem, thus suggesting that a more intricate approach would be needed to prove better impossibility results.

In this chapter we prove the optimal bound of $1-1/e$ on the best approximation that a single-pass $\tilde{O}(n)$ space algorithm can achieve *even in the vertex arrival setting*. While the lower bounds from Chapter 3 follow from a construction of a distribution on inputs that consists of two parts and hence yields a two-party communication problem, here we obtain an improvement by constructing hard input sequences that consist of k parts instead of two, getting a lower bound that approaches $1-1/e$ for large k. This can be viewed as multi-party communication complexity of bipartite matching, but we choose to present our lower bound in different terms for simplicity. We note that extending the approach of Chapter 3 sto a multi-party setting requires a substantially different construction. We discuss the difficulties and our approach to overcoming them in section 4.1.

Upper bounds. We show that a simple algorithm based on fractional load balancing achieves the optimal $1 - 1/e$ approximation in a single pass and $1 - \frac{1}{\sqrt{2\pi k}} + o(k^{-1/2})$ approximation in k passes, improving upon the best known algorithms for this setting:

Theorem 69 *There exists an algorithm for approximating the maximum matching M in a bipartite graph $G = (P, Q, E)$ with the P side arriving in the stream to factor $1 - e^{-k}k^{k-1}/(k-1)! = 1 - \frac{1}{\sqrt{2\pi k}} + O(k^{-3/2})$ in k passes using $O(|P| + |Q|)$ space and $O(m)$ time per pass.*

Remark 70 *Note that our algorithm extends trivially to the case when vertices in P have integral capacities $B_u, u \in P$, corresponding to advertiser budgets.*

The gap-existence problem. In [21] the authors give an algorithm for the closely related *gap-existence* problem. In this problem the algorithm is given a bipartite graph $G = (A, I, E)$, where A is the set of advertisers with budgets $B_a, a \in A$ and I is the set of impressions. The graph is lopsided in the sense that $|I| \gg |A|$. A matching M is *complete* if $|M \cap \delta(i)| = 1$ for all $i \in I$ and $|M \cap \delta(a)| = B_a$ for all $a \in A$. The gap-existence problem consists of distinguishing between two cases:

(YES) there exists a complete matching with budgets B_a;

(NO) there does not exist a complete matching with budgets $\lfloor (1-\epsilon)B_a \rfloor$.

The approach of [21] is via sampling the I side of the graph, and yields a solution that allows for non-trivial subsampling when the budgets are large. In particular, they obtain an algorithm with runtime $O\left(\frac{|A|\log|A|}{\epsilon^2} \cdot \frac{|I|}{\min_a |B_a|}\right)$, which is sublinear in the size of the graph when all budgets are large. In section 4.4 we improve significantly upon their result, showing

Theorem 71 *Gap-Existence can be solved in $O(\log(|I|\cdot\sum_{a\in B_a} B_a)/\epsilon^2)$ passes using space $O(\sum_{a\in A} B_a/\epsilon)$. The time taken for each pass is linear in the representation of the graph.*

It should also be noted that the result of [21] could be viewed as a single pass algorithm, albeit with the stronger assumption that the arrival order in the stream is random.

Organization: In section 4.1 we present the framework of our lower bound, which relies on a special family of graphs that we refer to as (d, k, δ)-packing. We then give a construction of a (d, k, δ)-packing in section 4.2. Our basic multipass algorithm for approximating matchings is presented in section 4.3, and the algorithm for Gap-existence is given in section 4.4.

4.1 Single pass lower bound

In this section we define the notion of a (d, k, δ)-*packing*, our main tool in proving the lower bound. A (d, k, δ)-packing is a family of graphs parameterized by the set of root to leaf paths in a d-ary tree of height k, inspired by Ruzsa-Szemerédi graphs, i.e. graphs whose edge set can be partitioned into large *induced* matchings. In this section we will show that existence of a (d, k, δ)-packing with a large number of edges implies lower bounds on the space complexity of achieving a better than $1 - 1/e$ approximation to maximum matchings in a single pass over the stream.

We first recall the definition of induced matchings and ϵ-Ruzsa-Szemerédi graphs.

Definition 72 *Let $G = (P, Q, E)$ denote a bipartite graph. A matching $F \subseteq E$ that matches a set $A \subseteq P$ to a subset $B \subseteq Q$ is* induced *if $E \cap (A \times B) = F$.*

Definition 73 *A bipartite graph $G = (P, Q, E)$ with $|P| = |Q| = n$ is an ϵ-Ruzsa-Szemerédi graph if one can write $E = \bigcup_{i=1}^{k} M_i$, where each M_i is an induced matching and $|M_i| = \epsilon n$ for all i.*

Several constructions of Ruzsa-Szemerédi graphs with a large number of edges are known. We will use the techniques pioneered in [29], where the authors construct ϵ-Ruzsa-Szemerédi graphs with constant $\epsilon < 1/3$, and the extensions developed in Chapter 3, where we proved that

Theorem 74 *[35] For any constant $\delta \in (0, 1/2)$ there exist bipartite $(1/2-\delta)$-Ruzsa-Szemerédi graphs on $2n$ nodes with $n^{1+\Omega(1/\log\log n)}$ edges.*

In the rest of the section we define a distribution on input instances for our problem of approximating maximum matchings in a single pass in the streaming model. We start by providing intuition for our distribution. It is useful to first recall how the lower bound of 3/4 for the same setting was proved in Chapter 3 (see section 3.7). The stream consists of two 'phases'. In the first phase, the algorithm is presented with a graph $G = (P, Q, E)$ such that $|P| = n, |Q| = 2n$ and the edge set E can be represented as a union of induced 2-matchings M_i, $i = 1, \ldots, k, k = n^{\Omega(1/\log\log n)}$, where M_i matches a subset $A_i \subseteq P$ such that $|A_i| \geq (1/2 - \delta)n$ to a subset $B_i \subseteq Q, |B_i| = (1 \pm \delta)n$. Then an index i is chosen uniformly at random from $[1 : k]$, and in the second part of the stream a matching arrives that matches a new set of vertices P^* to $Q^* = Q \setminus B_i$, making the edges of the (uniformly random) matching M_i crucial for constructing a better than 3/4 approximation to the maximum matching in the whole instance. It is then shown, using an additional randomization trick, that the algorithm essentially needs to store $\Omega(1)$ bits for each edge in each induced matching M_i if it beats the 3/4 approximation ratio.

We generalize this approach by constructing hard distributions on inputs that consist of *multiple phases*, for which any algorithm that achieves a better than $1 - 1/e$ approximation is essentially forced to remember $\Omega(1)$ bits per edge of the input graph. Ensuring that this is the case is the main challenge in generalizing the construction to a multiphase setting. We address this challenge using the notion of a (d, k, δ)-packing, which we now define.

4.1.1 (d, k, δ)-packing

Let $\mathcal{T}$ denote a d-ary tree of height k. A (d, k, δ)-packing will be defined as a function mapping root-to-leaf paths p in $\mathcal{T}$ to bipartite graphs on the vertex set (T, S), where T and S are the two sides of the bipartition. We will write $G(p)$ to denote the graph that a path p is mapped to by the packing.

The vertex set of $G(p)$ for each root-to-leaf path p will always be (T, S), so that the choice of p determines the set of edges of the graph. We partition the set S as $S = S_0 \cup \ldots \cup S_{k-1} \cup S_k$ (the sets $S_i, i = 0, \ldots, k$ are disjoint and correspond to $k + 1$ 'phases' of the input instance). We will always have $|T| = (1 + O(\delta))|S|$ for an arbitrarily small constant $\delta > 0$.

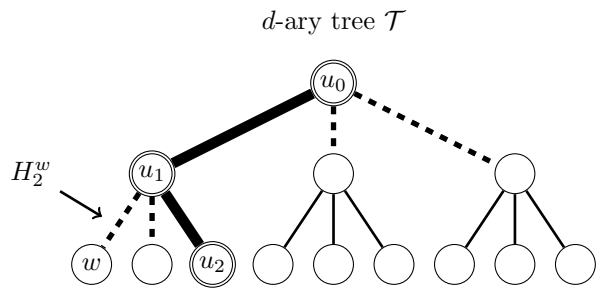


Figure 4.1: A root to leaf path in $\mathcal{T}$. Thick solid edges represent the edges of the path $(r = u_0, u_1, u_2)$. Dashed edges incident on nodes on the path $\mathcal{P}$ correspond to subgraphs H_i^w for $i = 0, 1$ and w a child of u_i.

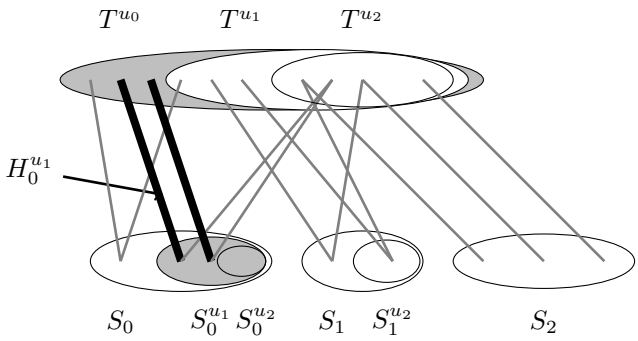


Figure 4.2: Subgraphs (T^{u_i}, S_i) that arrive in the stream. The edges of induced near-regular subgraph $H_0^{u_1}$ induced by $(T^{u_0} \setminus T^{u_1}) \cup S_0^{u_1}$ are shown in bold.

We now associate several sets of vertices on the T and S side of the bipartition with each node in the binary tree $\mathcal{T}$. Let $u \in \mathcal{T}$ be a node *at distance* $i \in [0, k]$ from the root. The following sets are associated with u:

1. a subset $T^u \subseteq T$, such that if w is a child of u in $\mathcal{T}$, one always has $T^w \subset T^u$;
2. for each $j \in [0 : i-1]$, a set S_j^u, such that if w is a child of u in $\mathcal{T}$, one always has $S_j^w \subset S_j^u$. To simplify notation, we set $S_i^{u_i} := S_i$.

We now describe the hard inputs that we will use. The input sequence is split into $k+1$ phases. The i-th phase corresponding to the i-th vertex on the path p from root to a leaf, where $i = 0, \ldots, k$ (see Fig. 4.1). During phase i the edges of the subgraph induced by $G_i(p) = (T^{u_i}, S_i, E_i(p))$ arrive in the stream. Crucially, the graph $G_i(p)$ will be a union of induced sparse subgraphs indexed by children of u_i.

This setup is illustrated in Fig. 4.1, where (a) all edges of the path $p = (r = u_0, u_1, u_2)$ are shown in bold and (b) all edges of $\mathcal{T}$ that are incident on nodes of p are dashed since the corresponding

subgraphs H_i^w arrive in the stream. The path p yields a nested sequence $T = T^{u_0} \supset T^{u_1} \supset \ldots T^{u_k}$ shown in Fig. 4.2.

The reason behind the fact that this construction presents a hard instance for small space algorithms is as follows. At each step i the algorithm is presented with all the subgraphs H_i^w, of which all except the uniformly random one (corresponding to the next node on the path p, i.e. $H_i^{u_{i+1}}$) will be useful for constructing a large matching in the whole instance. Large here means a matching of size at least a $(1-(1-1/k)^k+\delta')$ fraction of the maximum for some constant $\delta' > 0$. To show that only these special subgraphs are useful for constructing a large matching, we will later exhibit a *directed cut* of appropriate size in the graph $G(p)$ that consists only of the edges of $H_i^{u_{i+1}}$, $i = 0, \ldots, d-1$ (see Lemma 80). The key to exhibiting such a cut is the special structure of the sets $S_i^{u_k}$ for $i = 0, \ldots, k-1$ that we define in property (2) of (d,k,δ)-packings below. An additional randomization trick will allow us to show that a construction of a (d,k,δ)-packing immediately yields a lower bound of essentially $\Omega(dn)$ on the space required for a single-pass algorithm to achieve an approximation ratio better than $1-(1-1/k)^k+\delta'$ for a constant $\delta' > 0$.

We now transform the intuitive description above into a formal argument. We will use the following

Definition 75 *We call a bipartite graph $G = (P, Q, E)$ (a,b,δ)-*almost regular *if (1) at most a δ fraction of vertices in P has degree outside of $[(1-\delta)a, (1+\delta)a]$, and no vertex has degree larger than $(1+\delta)a$ and (2) at most a δ fraction of vertices in Q has degree outside of $[(1-\delta)b, (1+\delta)b]$, and no vertex has degree larger than $(1+\delta)b$.*

Definition 76 ((d,k,δ)-packing) *A mapping from the set of root-to-leaf paths p in a d-ary tree $\mathcal{T}$ to the set of bipartite graphs $G(p) = (T, S, E(p))$ is a (d,k,δ)-packing if the following conditions are satisfied.*

Let $p = (r = u_0, u_1, \ldots, u_k)$ be a root-to-leaf path in $\mathcal{T}$. Let $G(p) = (T, S, E(p))$ denote the graph that the path p is mapped to. Then the nested sequences of sets $T = T^{u_0} \supset T^{u_1} \supset \ldots \supset T^{u_{k-1}} \supset T^{u_k}$, and $S_i = S_i^{u_i} \supset S_i^{u_{i+1}} \supset \ldots \supset S_i^{u_k}$ satisfies the following properties for all $i = 0, \ldots, k-1$:

1. *For a constant $\gamma > 0$, one has for every child w of u_i in the tree $\mathcal{T}$ that the subgraph H_i^w induced by $(T^{u_i} \setminus T^w) \cup S_i^w$ is $((k-1)\gamma, k\gamma, \delta)$-almost regular.*
2. *there exists a set $Z^{u_i} \subset T$ such that $|Z^{u_i}| \leq O(\delta/k^2)|T^{u_i}|$, and the subgraph induced by $(T^{u_i} \setminus (T^{u_k} \cup Z^{u_i})) \cup S_i^{u_k}$ contains only the edges of $H_i^{u_{i+1}}$.*
3. *there exists a matching of at least a $1-\delta$ fraction of S_i to $T^{u_i} \setminus T^{u_{i+1}}$;*
4. *$|T^{u_i}| = (1+O(\delta))(1-1/k)^{-k+i}n$ and $|S_i^{u_j}| = (1+O(\delta))(1-1/k)^{-k+j}n/k$ for all $j = i, \ldots, k-1$.*
5. *there exists a matching of at least a $1-\delta$ fraction of S_k to T^{u_k}.*

Furthermore, for each $i = 0, \ldots, k$ the edge set of the subgraph induced by $T^{u_i} \cup S_i$ only depends on the nodes of p at distance at most i from the root.

Remark 77 *One could replace property (1) with the requirement that H_i^w be a matching of a $1-O(\delta)$ fraction of S_i^w to $T^{u_i} \setminus T^w$, and still get a lower bound that tends to $1-1/e$ for large k, albeit with slightly worse convergence. We prefer to use the more complicated definition to obtain the clean approximation ratio $1-(1-1/k)^k+O(\delta)$, where δ can be chosen an arbitrarily small constant, for any $k>1$.*

In what follows we will often refer to properties of (d,k,δ)-packings by number, without specifying each time that Definition 76 is meant.

In the rest of this section we will show that existence of large (d,k,δ)-packings implies space lower bounds for approximating matchings in one pass in the streaming model, thus proving

Theorem 78 *If a (d,k,δ)-packing with $\Theta(n)$ vertices exists for sufficiently large constant $k>0$ and $\delta=O(1/k^3)$, then no one-pass streaming algorithm can obtain a better than $(1-(1-1/k)^k+\delta')$-approximation for any constant $\delta'>0$ in space $o(nd)$, even when vertices on one side of the graph arrive in the stream together with all their edges.*

Together with the construction of a (d,k,δ)-packing with $d=n^{\Omega(1/\log\log n)}$ and $\delta=O(1/k^3)$ given in section 4.2, this will yield a proof of Theorem 68.

4.1.2 Distribution over inputs

We now formally define the (random) input graph $\mathcal{I}=(P,Q,E)$ based on a (d,k,δ)-packing. We will always have $P=\bigcup_{i=0}^{k} S_i$ and $Q=T$, but it will be useful to have notation for the parts P and Q of the bipartition of $\mathcal{I}$. Let $p=(r=u_0,u_1,\ldots,u_k)$ denote the path from the root of $\mathcal{T}$ to a uniformly random leaf. Let $G=G(p)$ denote the graph that the path p is mapped to by our (d,k,δ)-packing.

Let $T=T^{u_0}\supset T^{u_1}\supset T^{u_2}\supset\ldots\supset T^{u_k}$ denote the sequence of subsets of T corresponding to p. For each $i=0,\ldots,k-1$ and each child w of u_i let $H_i^w=(X_i^w,Y_i^w,E_i^w)$ denote the almost regular graph induced by $X_i^w\cup Y_i^w$, where $X_i^w=T^{u_i}\setminus T^w$ and $Y_i^w=S_i^w$.

We now introduce some randomness into the graph H_i^w. Let $\bar{H}_i^w$ be obtained from H_i^w via the following subsampling process. For each i and w let K_i^w denote a uniformly random subset of X_i^w of size $\delta|X_i^w|$ for a small constant δ. Let $b_x^{i,w}=1$ if $x\in K_i^w$ and 0 o.w. Then for each $x\in X_i^w$ the graph $\bar{H}_i^w$ contains all edges incident on x in H_i^w if $b_x^{i,w}=0$ and none of the edges incident on x otherwise. For each $i=0,\ldots,k-1$ let $\mathbf{b}_i=(b_x^{i,u_{i+1}})_{x\in X_{u_{i+1}}^i}$. Note that $\bar{H}_i^w$ is a $((k-1)\gamma,k\gamma,O(\delta))$-almost regular. For each $i=0,\ldots,k-1$ let $\bar{G}_i(p)=(T^{u_i},S_i,\bar{E}_i(p))$ denote the subgraph with bipartition (T^{u_i},S_i) such that $\bar{E}_i(p)$ is the union of the edges of all graphs $\bar{H}_i^w$ over all children w of u_i. Let $G_k(p)=(T^{u_k},S_k,\bar{E}_k(p))$ be a subgraph that consists of a perfect matching between S_k and T^{u_k} (see Fig. 4.2). The instance $\mathcal{I}$ is the union of $\bar{G}_i(p)$ over $i=0,\ldots,k$.

We now specify the order in which the vertices appear in the stream. The stream will consist of $k+1$ *phases*. For each $i=0,\ldots,k$ the vertices and edges of $\bar{G}_i(p)$ arrive in phase i in an arbitrary order.

This completes the description of the input. We now turn to proving Theorem 78. We will need the following claim

Claim 79 *G contains a matching of size at least $(1-O(\delta))(1-1/k)^{-k}n$.*

Proof: It is sufficient to match a $1-\delta$ fraction of S_i to $T^{u_i} \setminus T^{u_{i+1}}$ for all $i = 0, \ldots, k-1$, as guaranteed by property (3), and match the vertices in T^{u_k} to S_k. This matches a $1-O(\delta)$ fraction of T, and hence yields the required matching. ■

4.1.3 Bounding performance of a small space algorithm

By Yao's minimax principle it is sufficient to upper bound the performance of a deterministic small space algorithm that succeeds with probability at least $1/2$. To do that, we bound the size of the matching that a small space algorithm can output at the end of the stream. Let E^* denote the set of edges that an algorithm outputs at the end of the stream. We first upper bound the approximation ratio that the algorithm obtains in terms of the number of edges in $E(H_i^{u_{i+1}}) \cap E^*$, where $p = (u_0, u_1, \ldots, u_k)$ is the uniformly random path from the root to a leaf in $\mathcal{T}$.

Lemma 80 *The size of the matching output by the algorithm is bounded by*

$$\left((1-1/k)^{-k} - 1\right) n + \sum_{i=0}^{k-1} |E(H_i^{u_{i+1}}) \cap E^*| + O(\delta k^2 n).$$

Proof: Consider the cut (A, B), where $A = \left(T^0 \setminus (T^{u_k} \cup \bigcup_{i=0}^{k-1} Z^{u_i})\right) \cup \bigcup_{i=0}^{k-1} (S_i \setminus S_i^{u_k})$ and $B = T^{u_k} \cup S^* \cup \bigcup_{i=0}^{k-1} S_i^{u_k} \cup \bigcup_{i=0}^{k-1} Z^{u_i}$. Here Z^{u_i} are the sets whose existence is guaranteed by property (2).

By the maxflow/mincut theorem, the size of the matching output by the algorithm is bounded by $|A \cap P| + |B \cap Q| + |((A \cap Q) \times (B \cap P)) \cap E^*|$. Furthermore, by property (2) in Definition 76 for the sets A and B one has, using the fact that there are no edges from S^* to $T \setminus T^{u_k}$ that $((A \cap Q) \times (B \cap P)) \cap E \subset \bigcup_{i=0}^{k-1} E(H_i^{u_{i+1}})$, and hence

$$|((A \cap Q) \times (B \cap P)) \cap E^*| \leq \sum_{i=0}^{k-1} |E(H_i^{u_{i+1}}) \cap E^*|. \tag{4.1}$$

Combining these estimates, we get that the size of the matching output by the algorithm is bounded by

$$\left|\bigcup_{i=0}^{k-1} (S_i \setminus S_i^{u_k})\right| + |T^{u_k}| + \sum_{i=0}^{k-1} |Z^{u_i}| + \sum_{i=0}^{k-1} |E(H_i^{u_{i+1}}) \cap E^*|,$$

Recall that $|S_i| = (1+O(\delta))(1-1/k)^{-k+i}$ and $|S_i^{u_k}| = (1+O(\delta))n/k$ by property (4). Thus, the first term is at most

$$(1+O(\delta)) \left(\sum_{i=0}^{k-1} (1-1/k)^{-k+i} - 1\right) n/k = (1+O(\delta)) \left((1-1/k)^{-k} \frac{1-(1-1/k)^k}{1-(1-1/k)} - k\right) n/k$$
$$= (1+O(\delta))((1-1/k)^{-k} - 2)n.$$

Recalling that $T^{u_k} = (1+O(\delta))n$ by property (4) and $|Z^{u_i}| = O(k\delta)n$ by property (2) completes the proof. ■

We now show that no small space algorithm that is correct with probability at least $1/2$ can output more than a vanishingly small fraction of edges in $\bigcup_{i=0}^{k-1} E(H_i^{u_{i+1}})$. Recall that the vectors of bits flipped in the subsampling process that correspond to vertices (and their edge neighborhoods) in $\bar{H}_i^{u_{i+1}}$ are denoted by $\mathbf{b}_i$.

Lemma 81 *Let $\mathcal{I}$ denote the distribution on input graphs obtained from a (d,k,δ)-packing for constant k and $\delta = O(1/k^3)$. Let A be a $o(nd)$ space algorithm that is correct with probability at least $1/2$. Then for each $i = 0,\ldots,k-1$ the expected number of edges in $E(\bigcup_{i=0}^{k-1} H_i^{u_{i+1}})$ retained by A conditional on A being correct is $o(n)$.*

Proof:

We give the algorithm the following information for free. At the end of phase i the algorithm knows all vectors $\mathbf{u}_1,\ldots,\mathbf{u}_i$ on the path chosen in the distribution (of course, the algorithm does not know $\mathbf{u}_{i+1}$). This only makes the algorithm more powerful.

Let $\mathcal{G}_i$ denote the set of phase i graphs, i.e. the set of possible graphs on the vertices $T^{u_i} \cup S_i$. Since the algorithm knows all vectors $\mathbf{u}_1,\ldots,\mathbf{u}_i$, these graphs are solely determined by the choices made in the subsampling process in H_i^w for each w. Denote the state of the memory of the algorithm after i-th phase for $i = 0,\ldots,k-1$ by m_i. For each i between 0 and $k-1$ we denote the function that maps m_{i-1} and the graph $G_i = (T^{u_i}, S_i, E_i) \in \mathcal{G}_i$ to m_i by $\phi_i : \{0,1\}^s \times \mathcal{G}_i \to \{0,1\}^s$, where s is the number of bits of space that the algorithm uses. Wlog assume $m_{-1} = 0$.

Denote by E^* the set of edges that the algorithm outputs at the end of the stream. Denote the event that the algorithm is correct by $\mathcal{C}$. Let $E_i^* := E^* \cap (S_i \times Q)$. Let $M_i \in \{0,1\}^s$ denote the (random) state of the memory of the algorithm at the end of phase i. Let $\mathcal{D} := \{|E^*| = \Omega(n)\} \wedge \mathcal{C}$ and $\mathcal{D}_i = \{|E_i^*| = \Omega(1/k)n\} \wedge \mathcal{C}$.

We prove the lemma by contradiction. Suppose that conditional on being correct, the algorithm retains $\Omega(n)$ edges of $\bigcup_{i=0}^{k-1} E(H_i^{u_{i+1}})$. Then a simple averaging argument using the assumption that $\mathbf{Pr}[\mathcal{C}] \geq 1/2$ shows that $\mathbf{Pr}[\mathcal{D}] = \Omega(1)$ and there exists $j \in [0:k-1]$ such that $\mathbf{Pr}[\mathcal{D}_j] \geq C/k$ for a constant $C > 0$. We will now concentrate on phase j. Denote the set of *good* memory configurations by $G = \{(m_{j-1}, m_j) \in \{0,1\}^s : \mathbf{Pr}[\mathcal{D}_j | M_{j-1} = m_{j-1}, M_j = m_j] \geq C/(2k))\}$. Thus, G is a set of memory configurations in the $j-1$-st and j-th phases such that conditional on $(M_{j-1}, M_j) \in G$ the algorithm is likely to output a lot of edges of $\bigcup_{i=0}^{k-1} E(H_i^{u_{i+1}})$. Then

$$\mathbf{Pr}[(M_{j-1}, M_j) \in G] + (C/(2k))\mathbf{Pr}[(M_{j-1}, M_j) \notin G] \geq \mathbf{Pr}[\mathcal{D}_j] \geq C/k,$$

so

$$\mathbf{Pr}[(M_{j-1}, M_j) \in G] \geq C/(2k). \tag{4.2}$$

Before proceeding, we prove an auxiliary lemma. Recall that in the definition of a (d,k,δ)-packing for all d children w of u_i the graph $\bar{H}_i^w$ is obtained from H_i^w by keeping edges incident to a uniformly random subset of a $1-\delta$ fraction of nodes in X_i^w. Thus, there are at least $\binom{|X_i^w|}{\delta|X_i^w|}^d = 2^{\eta dn}$

graphs in $\mathcal{G}_i$, where $\eta > 0$ is a constant. The following claim follows similarly to Claim 54 (see section 3.7). We give a proof here for completeness.

Claim 82 *Let $\alpha > 0$ be a constant and let F be any subset of $\mathcal{G}_i$. Let G_F denote a set of edges that are contained in at least $1/2$ of the graphs in F. Let $J \subseteq [1:d]$ be the set of indices such that G_F contains at least $\alpha|X_i^w|$ edges from H_{i-1}^w, where w is the j-th child of u_{i-1}, for each $j \in J$. Then if $|F| \geq 2^{(\eta - o(1))dn}$, $|J| = o(d)$.*

Proof: Let $|J| = d_1$. Recall that by property (1) the maximum degree in H_i^w is bounded above by $c := (1 + O(\delta))\gamma k$. Thus, the number of graphs that can be in F is bounded by

$$\binom{(1-\alpha/c)|X_i^w|}{\delta|X_i^w|}^{d_1}\binom{|X_i^w|}{\delta|X_i^w|}^{d-d_1} = \left(2^{-\Omega(|X_i^w|)}\binom{|X_i^w|}{\delta|X_i^w|}\right)^{d_1}\binom{|X_i^w|}{\delta|X_i^w|}^{d-d_1} = 2^{-\Omega(d_1 n)}2^{\eta dn}.$$

It then follows that if $d_1 = \Omega(d)$, we have $|F| \leq 2^{(\eta - \Omega(1))dn}$, contradicting our assumption on the size of F. ■

Let $\mathcal{E}_j$ denote the event that $|\phi_{M_{j-1}}^{-1}(M_j)| \geq 2^{(\eta - o(1))dn}$. A simple counting argument shows that for a uniformly random graph $H \in \mathcal{G}_j$ we have $\mathbf{Pr}[\bar{\mathcal{E}}_j] = o(1)$ (here we use the fact that coin flips that determine which edges belong to H_{i-1}^w are independent of M_{j-1}). Combining this with (4.2), we get

$$\mathbf{Pr}[(M_{j-1}, M_j) \notin G] + \mathbf{Pr}[\bar{\mathcal{E}}_j] \leq 1 - C/(2k) + o(1) < 1.$$

Thus, there exists $m_{j-1}^*, m_j^* \in \{0,1\}^s$ such that the following properties hold

(P1) $\mathbf{Pr}[\mathcal{D}_j | M_{j-1} = m_{j-1}^*, M_j = m_j^*] \geq C/k$;

(P2) $|\phi_{m_{j-1}^*}^{-1}(m_j^*)| \geq 2^{(\eta - o(1))dn}$.

We can now complete the proof. For brevity let $\mathcal{M} = \{M_{j-1} = m_{j-1}^*, M_j = m_j^*\}$. Recall that E_j^* is the set of edges from $H_j^{u_{j+1}}$ that the algorithm outputs at the end of the stream. We have

$$\mathbf{E}_{E_j^*}\left[\mathbf{Pr}[\mathcal{D}_j | \mathcal{M}]\right] \geq C/k,$$

and so there exists $\hat{E}_j^*$ such that $\mathbf{Pr}[\mathcal{D}_j | \mathcal{M} \wedge E_j^* = \hat{E}_j^*] \geq C/k$.

Now recall that $\mathcal{D}_j = \{|E_j^*| = \Omega(1/k)n\} \wedge \mathcal{C}$. Thus, we have isolated memory configurations m_{j-1}^* and m_j^* and a set of edges $\hat{E}_j^*$ of size $\Omega(1/k)n$ such that the algorithm can output $\hat{E}_j^*$ and be correct with probability at least C/k conditional on $M_{j-1} = m_{j-1}^*$ and $M_j = m_j^*$!

Finally, note that conditional on $\mathcal{M} \wedge \{E_j^* = \hat{E}_j^*\}$ all graphs $H \in \phi_i^{-1}(m^*)$ are equiprobable. Now using property **P2** above together with Claim 82 we conclude that $|\hat{E}_j^*| = o(n)$, which is a contradiction.

■

We can now give

Proof of Theorem 78: The proof of Theorem 78 now follows by combining Claim 79, Lemma 80 and Lemma 81 after setting $\delta = c\delta'/k^2$ for a small constant $c > 0$. ■

4.2 Construction of a (d, k, δ)-packing

In this section we give a construction of a (d, k, δ)-packing on $\Theta(n)$ nodes with $d = n^{\Omega(\frac{1}{\log\log n})}$ for any constant k and sufficiently small constant $\delta > 0$. Our construction will use many of the techniques introduced in [29] and extensions of these techniques obtained in Chapter 3.

We first introduce notation. As before, the sides of the bipartition of the graph $G(p)$ that we need to construct are denoted by T and $S = S_0 \cup \ldots \cup S_k$. We use the notation $[a] = \{1, \ldots, a\}$ for integer $a \geq 1$. In our construction the $T = T^0$ side of the graph is identified with a hypercube $[m^4]^m$ for a value of m to be chosen later, and the sets $S_i, i = 0, \ldots, k-1$ are identified with a subsampled version of the hypercube $[m^4]^m$. The vertices of the last set S_k do not have any special structure. Vertices $x \in T$ or $y \in S_i$ will often be treated as points $x, y \in [m^4]^m$. Each node u of $\mathcal{T}$ (except the root) will be labeled with a binary vector $\mathbf{u} \in \{0,1\}^m$. We will write $|\mathbf{u}|$ to denote the Hamming weight of $\mathbf{u}$. For $x \in T$ and $u \in \mathcal{T}$ we use the dot product notation $(x, \mathbf{u}) = \sum_{i=1}^{m} x_i \cdot \mathbf{u}_i \in \mathbb{Z}$. For an interval $[a, b]$, where a, b are integers, and an integer number W we will write $[a, b] \cdot W$ to denote the interval $[a \cdot W, b \cdot W]$. Finally, for an integer i and an integer W we will write $i \mod W$ to denote the residue of i modulo W that belongs to $[0, W-1]$.

For convenience of the reader, we first give an informal outline of the construction. Given a path $p = (u_0, u_1, \ldots, u_k)$ from the root of $\mathcal{T}$ to a uniformly random leaf, we construct the packing as follows. First, we associate with each node of $\mathcal{T}$ other than the root a subset of $\{0,1\}^m$ (i.e. a binary vector) from a family of subsets of fixed cardinality and with small intersections. Since the subsets corresponding to nodes of $\mathcal{T}$ have small intersections, one can think of them as nearly orthogonal vectors.

We then traverse the path p from the root to the leaf and at step i, $i = 0, \ldots, k-1$ we essentially set[1]

$$T^{u_{i+1}} := \{x \in T^{u_i} : (x, \mathbf{u}_{i+1}) \mod W \in [1/k, 1] \cdot W\},$$

where W is an appropriately chosen parameter. Thus, traversing a root to leaf path amounts to repeatedly cutting the hypercube with hyperplanes whose normal vectors are almost orthogonal. At step i the set S_i is identified with an appropriately subsampled copy of T^{u_i}, and a Ruzsa-Szemerédi graph is constructed on (T^{u_i}, S_i). At step i, besides defining the new set $T^{u_{i+1}}$, the vector $\mathbf{u}_{i+1}$ (corresponding to the next vertex on the path) is used to define a subset $S_j^{u_{i+1}} \subseteq S_j^{u_i}$ for all $j \leq i$ by similarly cutting $S_j^{u_i}$ with a hyperplane. The most important property of our construction will be the fact that when we reach the leaf u_k, most of the edges going out of $S_j^{u_k}$ for $j = 0, \ldots, k-1$ will be contained in T^{u_k}, yielding property (2) of (d, k, δ)-packings. Our main contribution here is the approach of constructing a recursive sequence of graphs by cutting the hypercube by nearly orthogonal hyperplanes, which allows us to derive property (2).

We now give the details of the construction. We will use the following lemma, whose proof is implicit in the proofs of lower bounds from Chapter 3 (see section 3.7), which is a convenient formulation of the construction of error correcting codes with fixed weight in [66]

[1] This statement is slightly imprecise in the interest of clarity.

Lemma 83 *[35] For sufficiently large $m > 0$, any constant $\epsilon \in (0,1)$ and constant $\gamma \in (0,2)$ there exists a family $\mathcal{F}$ of subsets of $[m]$ of size ϵm with intersection at most $\gamma\epsilon^2 m$ such that $\frac{1}{m}\log|\mathcal{F}| \geq c_{\epsilon,\gamma} - o(1)$.*

Our main lemma is

Lemma 84 *For any constants $k, \delta' > 0$ there exists a (d, k, δ')-packing on $\Theta(n)$ nodes with $d = n^{\Omega(\frac{1}{\log\log n})}$.*

Proof: We associate with each node of the d-ary tree $\mathcal{T}$ of height k a vector $\mathbf{v}$ from a family of almost orthogonal binary vectors of equal weight whose existence is guaranteed by Lemma 83. Since the number of nodes in such a tree is at most d^{k+1}, we can afford to set $d = 2^{\Omega(m)}$ since k is constant. Besides associating with each node $u \in \mathcal{T}$ a vector $\mathbf{u}$, we also associate with u a random variable U_u that is uniformly distributed over the integers between 0 and $W-1$, where W is a parameter that will be chosen later. The variables U_u and $U_{u'}$ are independent for $u \neq u'$.

Let $X' = Y = [m^4]^m$ for some integer $m > 0$. Let X be a uniformly random subset of X' where each point of X' appears independently with probability $1/k$. We will refer to vertices in X and Y as points in $[m^4]^m$. We now specify how a graph satisfying the properties in definition 76 is constructed for a given path $p = (u_0, u_1, \ldots, u_k)$ denote a path from the root of $\mathcal{T}$ to a leaf of $\mathcal{T}$.

The path p induces a decomposition of the vertex set T as follows. For all $i = 0, \ldots, k-1$

$$\begin{aligned} T^{u_i} &= \{y \in Y : (y, \mathbf{u}_j) \mod W \in [1/k, 1) \cdot W, \text{ for all } j \in [1:i]\} \\ S_i &= \{x \in X' : (x, \mathbf{u}_j) \mod W \in [1/k, 1) \cdot W, \text{ for all } j = [1:i]\}. \end{aligned} \tag{4.3}$$

Also, let

$$S_j^{u_i} = \{x \in S_j : (x, \mathbf{u}_l) \mod W \in [1/k, 1) \cdot W, \text{ for all } l \in [1:i]\}, \text{ for all } j = 0, \ldots, i-1 \tag{4.4}$$

The set S_k is a disjoint set of vertices connected to T^{u_k} by a perfect matching.

Consider fixed i between 0 and $k-1$. For all children w of u_i let

$$\begin{aligned} R^Y(w) &= \{y \in T^{u_i} : ((y, \mathbf{w}) + U_w) \mod W \in [0, 1/k] \cdot W\} \\ W^Y(w) &= \{y \in T^{u_i} : ((y, \mathbf{w}) + U_w) \mod W \in ([1/k, 1/k + \delta] \cup [1 - \delta, 1)) \cdot W\} \\ B^Y(w) &= \{y \in T^{u_i} : ((y, \mathbf{w}) + U_w) \mod W \in [1/k + \delta, 1 - \delta] \cdot W\} \end{aligned} \tag{4.5}$$

Define $R^X(w), W^X(w), B^X(w)$ similarly (note that these sets are defined only for S_i):

$$\begin{aligned} R^X(w) &= \{x \in S_i : ((x, \mathbf{w}) + U_w) \mod W \in [0, 1/k] \cdot W\} \\ W^X(w) &= \{x \in S_i : ((x, \mathbf{w}) + U_w) \mod W \in ([1/k, 1/k + \delta] \cup [1 - \delta, 1)) \cdot W\} \\ B^X(w) &= \{x \in S_i : ((x, \mathbf{w}) + U_w) \mod W \in [1/k + \delta, 1 - \delta] \cdot W\} \end{aligned} \tag{4.6}$$

We note here that the random shift U_w is not necessary for most properties that we establish, and will only be useful to establishing property (3). First, we analyze

Size of the sets $T^{u_i}, S_j, S_j^{u_i}, R, B, W$ and property (4). We will need

Claim 85 *Let $\delta > 0$ be a constant such that $1/\delta$ and $\delta W/|\mathbf{w}|$ are integers, and let $U \in [0 : W-1]$ be an integer. Define for $q = 0, \ldots, 1/\delta - 1$*

$$A_q = |\{y \in Y : ((y, \mathbf{u}_j) + U) \mod W \in [\delta q, \delta(q+1)] \cdot W|\,. \tag{4.7}$$

Then $|A_q| \in (1 \pm o(1))\delta|Y|$.

Proof: Consider the mapping $\psi : y \to y - \frac{\delta W}{|\mathbf{u}_j|} \cdot \mathbf{u}_j$. This is a well defined mapping into Y for all $y \in Y$ except those that have at least one coordinate smaller than $\frac{\delta W}{|\mathbf{u}_j|} = O(1)$. We denote this set by R. But for any fixed l one has $|\{y \in Y : y_l < \frac{\delta W}{|\mathbf{u}_j|}\} = \frac{\delta W}{m^4 |\mathbf{u}_j|} = o(|Y|/m^2)$, and hence by the union bound over all $l = 1, \ldots, m$ one has $|R| = o(|Y|)$. For all $q = 1, \ldots, 1/\delta - 1$ the mapping ϕ maps A_q injectively into A_{q-1}, and A_0 into $A_{1/\delta - 1}$, everywhere except R. Thus, one has $|A_q| = \delta(1 \pm o(1))|Y|$, and the conclusion of the lemma follows. ■

We first prove

Lemma 86 *Consider any set $\mathcal{S}$ defined by $\mathcal{S} = \{y \in Y : (y, \mathbf{u}) \mod W \in [a_\mathbf{u}, b_\mathbf{u}] \cdot W, \mathbf{u} \in \mathcal{U}\}$, where $\mathcal{U}$ is a collection of binary vectors and $a_\mathbf{u}, b_\mathbf{u}$ are constants. Let $\mathbf{v}$ be a vector such that $|\mathbf{u}| = |\mathbf{v}|$ for all $\mathbf{u} \in \mathcal{U}$ and $\max_{\mathbf{u} \in \mathcal{U}}(\mathbf{u}, \mathbf{v})/|\mathbf{v}| \le \delta'$, and $A, B \in [0, 1], A \le B$ are rational constants. Let*

$$\mathcal{S}' = \{y \in \mathcal{S} : (y, \mathbf{v}) \mod W \in [A, B] \cdot W\}.$$

Then for sufficiently large $W = O(m)$ one has $||\mathcal{S}'| - (B - A)|\mathcal{S}|| = O(|\mathcal{U}|\delta')$.

Proof: Consider the mapping $\psi_{\mathbf{v},j} : y \to y - \frac{j \cdot \delta(B-A)W}{|\mathbf{v}|} \cdot \mathbf{v}$, where δ is a sufficiently small rational constant such that $1 - (B - A)$ is an integer multiple of $\delta(B - A)$. Note that the mapping is well-defined as long as W is an integer multiple of $1/(\delta(B - A))$, which is admissible under our assumption that $W = O(m)$.

Let $y \in \mathcal{S}$. Then

$$(\psi_{\mathbf{v},j}(y), \mathbf{u}) = (y, \mathbf{u}) + \frac{j \cdot \delta(B-A)W}{|\mathbf{v}|} \cdot (\mathbf{u}, \mathbf{v}) \le (y, \mathbf{u}) + j \cdot \delta(B - A)W\delta',$$

so $\psi_{\mathbf{v},j}$ for $|j| \le 1/(\delta(B - A))$ maps points $y \in \mathcal{S}$ into $\mathcal{S}$ unless either

$$(y, \mathbf{u}) \mod W \in [a_\mathbf{u}, a_\mathbf{u} + \delta'] \cup [b_\mathbf{u} - \delta', b_\mathbf{u}] \cdot W \tag{4.8}$$

for at least one $\mathbf{u} \in \mathcal{U}$ or y has at least one coordinate smaller than W. We call such points *bad* and denote this set by R. For a fixed $\mathbf{u}$ the fraction of $y \in Y$ that do not satisfy (4.8) is $O(\delta')$ by Claim 85 and hence by the union bound over all $\mathbf{u} \in \mathcal{U}$ we get that the fraction of such points in Y is $O(|\mathcal{U}|\delta')$. The fraction of points with at least one coordinate smaller than W is at most W/m^4, and hence by the union bound the fraction of points with at least one coordinate smaller than W is $o(1)$, so $|R| = O(\mathcal{U}\delta') \cdot |Y|$.

Similarly to Claim 85, define

$$A_q = |\{y \in \mathcal{S} : (y, \mathbf{v}) \mod W \in [(B-A)\delta q, (B-A)\delta(q+1)] \cdot W|\,. \tag{4.9}$$

Now let $D = [0 : \frac{1}{(B-A)\delta})$ denote the set of indices such that $\mathcal{S} = \bigcup_{d \in D} A_d$, and let $D' = [\frac{A}{(B-A)\delta} : \frac{B}{(B-A)\delta}]$ denote the set of indices such that $\mathcal{S}' = \bigcup_{d \in D'} A_d$.

Define a bipartite graph $F = (\mathcal{S}', \mathcal{S} \setminus \mathcal{S}', E_F)$ by including an edge $(x, y), x \in \mathcal{S}', y \in \mathcal{S} \setminus \mathcal{S}'$ to E_F whenever $\psi_{\mathbf{v},j}(x) = y$ for some $j \in D$. Thus, each $x \in \mathcal{S}' \setminus R$ has degree $|D \setminus D'|$ in F, and $x \in (\mathcal{S} \setminus \mathcal{S}') \setminus R$ have degree $|D'|$ in F. Furthermore, the degree of each $x \in \mathcal{S}'$ is bounded by $|D \setminus D'|$ and the degree of each $x \in \mathcal{S} \setminus \mathcal{S}'$ is bounded by $|D'|$.

Putting these estimates together, we have $|\mathcal{S}' \setminus R| \cdot |D \setminus D'| \leq |\mathcal{S} \setminus \mathcal{S}'| \cdot |D'|$, i.e.

$$|\mathcal{S}'| \leq (|\mathcal{S}| - |\mathcal{S}'|) \cdot \frac{|D'|}{|D \setminus D'|} + |R| = (|\mathcal{S}| - |\mathcal{S}'|) \cdot \frac{B-A}{1-(B-A)} + |R|.$$

Thus, $|\mathcal{S}'| \leq (B-A) \cdot |\mathcal{S}| + (1-(B-A))|R|$. On the other hand, we also have $|(\mathcal{S} \setminus \mathcal{S}') \setminus R| \cdot |D'| \leq |\mathcal{S}'| \cdot |D \setminus D'|$, i.e.

$$|\mathcal{S} \setminus \mathcal{S}'| \leq |\mathcal{S}'| \cdot \frac{|D \setminus D'|}{|D'|} + |R| = |\mathcal{S}'| \cdot \frac{1-(B-A)}{B-A} + |R|$$

Thus, $(B-A)(|\mathcal{S}| - |\mathcal{S}'|) \leq |\mathcal{S}'| \cdot (1-(B-A)) + (B-A)|R|$, so $|\mathcal{S}'| \geq (B-A)|\mathcal{S}| - (B-A)|R|$. The conclusion of the lemma follows. ■

Estimates on the size of sets T^{u_i} now follow by noting that one has $|\mathcal{U}| \leq k$ in all cases, and that the maximum dot product δ' can be chosen to be $1/\text{poly}(k)$. The bounds on the size of $S_i^{u_j}, R, B, W$ follow in a similar way with the additional application of Chernoff bounds to the sampling of points that are included in X'.

We now define the edges of the $((k-1)\gamma, k\gamma, O(\delta))$-almost regular induced subgraph H_i^w, for a constant $\gamma > 0$ (the induced property will be shown later). The subgraph H_i^w will consist of disjoint copies of small complete bipartite graphs.

Constructing H_i^w. Fix a child w of u_i. For the purposes of constructing H_i^w we *condition on the values of all shifts U_w*. In what follows we omit the parameter w when referring to sets $R^Y(w), W^Y(w), B^Y(w)$. For two vertices $b, b' \in R^Y$ such that $|(b - b', \mathbf{w})| \leq W/k$ we say that $b \sim b'$ if $b - b' = \lambda \cdot \mathbf{w}$ for some λ. Note that we have $\lambda \in \left[-\frac{W}{k|\mathbf{w}|}, \frac{W}{k|\mathbf{w}|}\right]$. We write $\mathcal{B}_b \subseteq Y$ to denote the equivalence class of b. It follows directly from the definition of $\mathcal{B}_b$ and (4.5) that $|\mathcal{B}_b| = W/(k|\mathbf{w}|)$ for all b. Also, let

$$\mathcal{A}_b = B^X \cap \left(\bigcup_{\lambda \in [0, (1-1/k)W/|\mathbf{w}|]} (\mathcal{B}_b + \lambda \cdot \mathbf{w}) \right).$$

Note that $\mathcal{A}_b$ is a random set (determined by the random choice of $X \subset X'$). Since each element of X' is included in X independently with probability $1/k$, we have that $\mathbf{E}[|\mathcal{A}_b|] = (1 \pm O(\delta))(1-1/k)|\mathcal{B}_b|$.

We now define a set of edges of a $((k-1)\gamma, k\gamma, \delta)$-almost regular subgraph between (a subset of) $\mathcal{B}_b$ and $\mathcal{A}_b$. First note that $\mathbf{E}[|\mathcal{B}_b|] = (1 \pm O(\delta))(1-1/k)|\mathcal{A}_b|$. Furthermore, since X is obtained from X' by independent sampling at rate $1/k$, standard concentration inequalities yield

$$\mathbf{Pr}\left[|\mathcal{A}_v| \notin (1 \pm \delta)(1-1/k)|\mathcal{B}_v|\right] \leq e^{-\delta^2(1/2)|\mathcal{B}_v|/4} \leq \delta^2 \tag{4.10}$$

for $|\mathcal{A}_b| > \gamma = 16\ln(8/\delta)/\delta^2$. To ensure this, it is sufficient to ensure that $W \geq \frac{16k\ln(8/\delta)}{\delta^2} \cdot |\mathbf{w}|$. We note here that we are thinking of δ as being smaller than $1/k$. In particular, we will set $\delta = O(1/\text{poly}(k))$ at the end of the construction. Now for each $c \in \mathcal{A}_b, d \in \mathcal{B}_b$ include an edge (c,d) in H_i^w. We will define a complete bipartite graph on each such equivalence class $\mathcal{A}_b, \mathcal{B}_b$, i.e. for each $c \in \mathcal{A}_b, d \in \mathcal{B}_b$ include an edge (c,d) in H_i^w. However, since we used randomness to chose the set X', some of these classes may be too small due to stochastic fluctuations. We deal with this problem next.

We now classify points $b \in R^Y$ as good or bad depending on the how close $|B_b|$ is to its expectation. In particular, mark a b *bad* if $|\mathcal{B}_b| \notin (1 \pm \delta)(1-1/k)|\mathcal{A}_b|$ and *good* otherwise. Note that in fact this is a well-defined property of an equivalence class. Let $J_{\mathcal{B}}$ denote the indicator random variable that equals 1 if $\mathcal{B}$ is bad and 0 otherwise, where $\mathcal{B}$ is an equivalence class. Note that $J_{\mathcal{B}}$ is independent of $J_{\mathcal{B}'}$ for $\mathcal{B} \neq \mathcal{B}'$, since J is determined by the random choice of $X \subset X'$ and we are conditioning on the values of all shifts $U_w, w \in \mathcal{T}$. By (4.10) one has $\mathbf{E}[J_{\mathcal{B}}] \leq \delta^2$ for all equivalence classes $\mathcal{B}$. Note that each equivalence class contains a constant number of points, and hence there are $\Omega(m^{4m})$ equivalence classes for every i and w child of u_i.

An application of Chernoff bounds shows that for fixed i and fixed w a child of u_i

$$\mathbf{Pr}\left[\sum_{\mathcal{B}} J_{\mathcal{B}} > 2\mathbf{E}\left[\sum_{\mathcal{B}} J_{\mathcal{B}}\right]\right] \leq e^{-\Omega(m^{4m})}. \tag{4.11}$$

Note that by (4.10) one has that (4.11) bounds the probability of there being more than $2\delta^2$ fraction of bad classes for fixed $w \in \mathcal{T}$. Taking a union bound over $2^{O(m)}$ nodes of $\mathcal{T}$, we conclude that there will be no more than $2\delta^2$ fraction of bad equivalence classes in H_i^w *for any* i*, and* w *a child of* u_i.

If b is good, let $\mathcal{A}'_b$ denote an arbitrary subset of $\mathcal{A}_b$ of cardinality $(1-\delta)(1-1/k)|\mathcal{B}_b|$. Similarly, let $\mathcal{B}'_b$ denote an arbitrary subset of $\mathcal{B}_b$ of cardinality $(1-\delta)|\mathcal{B}_b|$, so that $|\mathcal{A}'_b| = (1-1/k)|\mathcal{B}'_b|$. Now for each $c \in \mathcal{A}'_b, d \in \mathcal{B}'_b$ include an edge (c,d) in H_i^w. Note that each such graph is a $((k-1)\gamma, k\gamma, \delta)$-almost regular graph, as required by property (1). Note that all matched edges are of the form (c,d), where

$$c = d - \lambda \cdot \mathbf{w}, \lambda \in (0, W/|\mathbf{w}|]. \tag{4.12}$$

The union of the small complete graphs that we constructed yields the graph H_i^w for a fixed child w of u_i. We also showed that on such graph H_i^w contains more than a $2\delta^2$ fraction of bad classes whp, which completes the construction of the graphs H_i^w.

Induced property (property (1)). Graphs H_i^w constructed in this way are induced for the same reason as in the constructions in [29] and Chapter 3 when the vectors $\mathbf{w}, \mathbf{w}'$ corresponding to two distinct nodes of $\mathcal{T}$ are chosen in such a way that $|\mathbf{w}| = |\mathbf{w}'| = \epsilon m$ (recall that $X' = Y = [m^4]^m$) and

$$(\mathbf{w}, \mathbf{w}') \leq (5/2)\epsilon|\mathbf{w}| \tag{4.13}$$

for sufficiently small constant ϵ. Indeed, consider a fixed i and suppose that an edge $(a, b) \in E(H_i^w)$ is induced by $H_i^{w'}$ for $w' \neq w$. But then it must be that either $c \in R^Y(w'), d \in B^X(w')$ or $d \in R^Y(w'), c \in B^X(w')$. In either case one has

$$|(c - d, \mathbf{w}')| \geq \delta \cdot W. \tag{4.14}$$

However, by (4.13) together with (4.12) one has

$$|(c - d, \mathbf{w}')| \leq \frac{W}{|\mathbf{w}|}(\mathbf{w}, \mathbf{w}') \leq \frac{W}{|\mathbf{w}|}(5/2)\epsilon|\mathbf{w}| = (5/2)\epsilon W,$$

which is a contradiction with (4.14) for $\epsilon < \delta/10$.

Existence of a large matching (property (3)) We now show that for any i and w a child of u_i there exists a matching of $1 - O(\delta)$ fraction of S_i to $T^{u_i} \setminus T^w$. We will do this by exhibiting a fractional matching of appropriate size.

Consider a point $x \in T^{u_i}$. We need to analyze the degree of x in the graph $T^{u_i} \cup S_i$. Note that the degree of x depends on (1) the number of vectors w for which $x \in R^Y(w)$ and (2) on the size of the equivalence classes that x belongs to for different w. We first analyze (1).

For a fixed w it follows by Claim 85 and the definition of U_w that $\mathbf{Pr}_{U_w}[x \in R^Y(w)] \in (1 \pm o(1))\frac{1}{k}$. Next note that each vertex $x \in R^Y(w)$ has degree $(k-1)\gamma$ in H_i^w. Furthermore, since the random shifts U_w are independent for different w, we obtain using Chernoff bounds that for a fixed $x \in T^{u_i}$

$$\mathbf{Pr}\left[\sum_{w \text{ child of } u_i} \mathbf{1}_{x \in R^Y(w)} \notin (1 \pm O(\delta))d/k\right] \leq e^{-\Omega(\delta^2 d/k)}. \tag{4.15}$$

A similar argument shows that the expected degree of each vertex in $S_i \setminus S_i^w$ has similar concentration around $k\gamma d$. Since there are only $O(m^{4m})$ vertices and $2^{O(m)}$ nodes in the tree $\mathcal{T}$, and $d = 2^{\Omega(m)}$, a union bound shows that vertex degrees are concentrated in each T^{u_i}, S_i pair with high probability. Now it remains to handle the loss of edges due to $x \in T^{u_i}$ belonging to small equivalence classes for some w. However, it follows from the analysis in (4.11) that at most an $O(\delta^2)$ fraction of the edge mass can be lost because of this, yielding the following fractional matching. Put weight $1/(k\gamma)$ on each edge in H_i^w, and put weight $\frac{1}{(1+O(\delta))k(1-1/k)\gamma d}$ on each edge going from $T^{u_i} \setminus T^w$ to $S_i \setminus S_i^w$. Since degrees in T^{u_i} are bounded by $(1 + O(\delta))(1 - 1/k)\gamma d$, and degrees is S_i are bounded by $(1+O(\delta))k\gamma d$, this is feasible and yields a matching of size $(1-O(\delta+\delta^2))|S_i|$, proving property (3).

We now prove property (2). For $i = 0, \ldots, k-1$ let

$$Z^{u_i} = \{y \in Y : (y, \mathbf{u}_j) \mod W \in ([1/k - \delta, 1/k] \cup [0, \delta]) \cdot W \text{ for some } j \in [1:k]\}. \tag{4.16}$$

We need to show that the subgraph H^* induced by $(T^{u_i} \setminus (T^{u_k} \cup Z^{u_i})) \cup S_i^{u_k}$ only contains the edges of $H_i^{u_{i+1}}$. First note that if an edge $(c,d), c \in P, d \in Q$ belongs to H^*, then $c \in S_i^{u_k}$ and $d \in T^{u_i}$, so (c,d) necessarily belongs to some graph H_i^w, where w is a child of u_i. Then we have by (4.12) that

$$d - c = q \cdot \mathbf{w}, \text{ where } |q| \leq W/|\mathbf{w}|.$$

On the other hand, we have for all $j = 1, \ldots, k$ using the orthogonality condition (4.13)

$$|(c - d, \mathbf{u}_j)| \leq \frac{W}{|\mathbf{w}|} |(\mathbf{w}, \mathbf{u}_j)| \leq (5/2)\epsilon W. \tag{4.17}$$

Now recall that $a \in S_i^{u_k}$, so by (4.3) and (4.4)

$$(c, \mathbf{u}_j) \mod W \in [1/k, 1] \cdot W, \forall j = 1, \ldots, k.$$

Thus, by (4.17) one has

$$(d, \mathbf{u}_j) \mod W \in ([1/k - \delta, 1] \cup [0, \delta]) \cdot W, \forall j \leq k,$$

i.e. $d \in Z^{u_i} \cup T^{u_k}$, if we set ϵ to smaller than $\delta/10$.

It remains to bound the size of Z^{u_i}. First note that it follows from Claim 85 that for sufficiently small constant δ (e.g. $\delta < 1/k^2$) one has

$$|\{y \in Y : (y, \mathbf{u}_j) \mod W \in ([1/k - \delta, 1/k] \cup [0, \delta]) \cdot W| \leq 2\delta |Y|. \tag{4.18}$$

Now by a union bound over all $j \in [1:k]$ we conclude that $|Z^{u_i}| \leq 2\delta k |Y| = O(\delta k n)$.

It remains to set parameters. First, inspection of the bounds obtained so far reveals that setting $\delta = c\delta'/k^4$ for a sufficiently small constant $c > 0$ is sufficient to obtain a (d, k, δ')-packing, where we set $\epsilon = \delta/10$. Finally, the size of the graphs obtained is essentially the same as in [29] and our constructions in Chapter 3. In particular, the number of vertices is $n = \Theta(m^{4m})$ and $d = 2^{\Omega(m)}$. Thus, we get a graph on n vertices with $d = n^{\Omega\left(\frac{1}{\log \log n}\right)}$ edges. ■

Proof of Theorem 68: The proof follows by combining Theorem 78 and Lemma 84. ■

4.3 Multipass approximation for matchings

In this section we present the basic version of our algorithm for approximating matchings in multiple passes in the vertex arrival setting. Let $G = (P, Q, E)$ denote a bipartite graph. We assume that vertices in P arrive in the stream together with all their edges. At each step the algorithm maintains

a fractional matching $\{f_e\}_{e \in E}$, where the capacity of each vertex in Q is infinite and the capacity of each vertex $u \in P$ is equal to the number of times it has appeared in so far (i.e. always between 1 and k). The capacity of an edge $e = (u, v), u \in P, v \in Q$ is equal to the capacity of u. For a vertex $u \in P$ we write $\delta(u)$ to denote the set of neighbors of u in G.

4.3.1 Algorithm

We now give the algorithm and show how to implement each pass in linear time.

Algorithm 1: PROCESS-VERTEX(G, u, $\delta(u)$)

1: Augment capacity of u and all edges in $\delta(u)$ by 1.
2: WATER-FILLING$(G', u, \delta(u))$
3: REMOVE-CYCLES(G', f).

The function WATER-FILLING$(G', u, \delta(u))$ increases the load of the least loaded neighbors of u simultaneously (with other neighbors joining if the load reaches their level) until one unit of water is dispensed out of u. Here the support of the fractional matching $\{f_e\}_{e \in E}$ maintained by the algorithm is denoted by G'. The function REMOVE-CYCLES(G', f) reroutes flow among cycles that could have emerged in the process, ensuring that the flow is supported on at most $|P| + |Q| - 1$ edges. We note that as stated, Algorithm 1 does not necessarily take $O(m)$ time per pass due to the runtime of cycle removal. However, simply buffering incoming vertices until the number of edges received is $\Theta(n)$ and only then removing cycles yields a linear time implementation. Here we can use DFS to reroute flow along cycles in time linear in the number of nodes.

Remark 87 *We note that a single pass of this algorithm is different from the one-pass algorithm that achieves $1 - 1/e$ approximation from [35]. However, we will later show that our algorithm in fact also achieves the ratio of $1 - 1/e$ in a single pass.*

We now turn to analyzing the approximation ratio. We first give a sketch of the proof under additional assumptions on the graph G, and then proceed to give the relevant definitions and the complete argument.

4.3.2 Analysis for a simple case

We now assume that $G = (P, Q, E)$ has a perfect matching M. For each $k \geq 1$ and all $x \geq 0$ denote by $b^k(x)$ the number of vertices in Q that have load *at least* x after k passes. We start by pointing out some useful properties of the function $b^k(x)$. First, note that $b^k(0) = |M|$, $b^k(x)$ is non-increasing in x and $b^k(x) - b^{k-1}(x) \geq 0$ for all x. Furthermore, we have

$$\int_0^\infty b^k(x)dx = k|M|, \tag{4.19}$$

since every vertex $u \in P$ contributed 1 unit of water, amounting to $|M|$ amount of water overall, and (4.19) calculates the sum of loads on all $v \in Q$. Furthermore, note that the size of the matching

constructed by the algorithm after k passes is exactly equal to

$$\frac{1}{k}\int_0^k b^k(x)dx, \tag{4.20}$$

since every vertex $v \in Q$ with load x contributed $\frac{1}{k} \cdot \min\{k, x\}$ to the matching. Hence the approximation ratio after k passes is at least

$$1 - \frac{1}{k}\int_k^\infty b^k(x)dx, \tag{4.21}$$

where we used (4.19) to convert (4.20) into (4.21). Thus, it is sufficient to lower bound $\int_0^k b^k(x)dx$ in order to analyze the approximation ratio, and we turn to bounding this quantity.

First consider the case $k = 1$. Fix $x \geq 0$ and consider vertices $v \in Q$ that have load at least x – there are at least $\int_x^\infty b^1(s)ds$ of them. For each such vertex u consider its match $M(u)$. Since u ended up at level at least x after the first pass, its match $M(u)$ must have been at level at least x when u arrived, and levels are monotone increasing. Hence, we have

$$b^1(x) \geq \int_x^\infty b^1(s)ds \tag{4.22}$$

for all $x \geq 0$. This, however, together with (4.19) can be shown to imply that $\int_x^\infty b^1(s)ds \leq |M| \cdot e^{-x}$ for all x. We immediately get using (4.21) that the approximation ratio after one pass is at least $1 - 1/e$.

Now suppose that $k > 1$ and consider vertices $v \in Q$ that are at level at least x after k-th pass, but were at a lower level after $(k-1)$-st pass. There are exactly $b^k(x) - b^{k-1}(x)$ such vertices. Since these vertices u were at level at least x after k-th pass, their matches $M(u)$ must have also been at level at least x when they arrived, implying that

$$b^k(x) \geq \int_x^\infty (b^k(s) - b^{k-1}(s))ds \tag{4.23}$$

for all $x \geq 0$. Solving (4.23), we get that for all $k \geq 1$

$$\int_x^\infty b^k(s)ds \leq |M| \cdot \int_x^\infty F^k(s)ds, \tag{4.24}$$

where $1 - F^k(x)$ is the cdf of the Gamma distribution with scale 1 and shape k, i.e. $F^k(x) = \int_x^\infty e^{-s}s^{k-1}/(k-1)!ds$. Using this in (4.21) yields the desired bound on the approximation ratio, i.e. $1 - e^{-k}k^{k-1}/k!$.

4.3.3 General case

The proof sketch we gave in the previous subsection works under the assumption that G has a perfect matching. The general case turns out to be substantially more involved. Interestingly, while

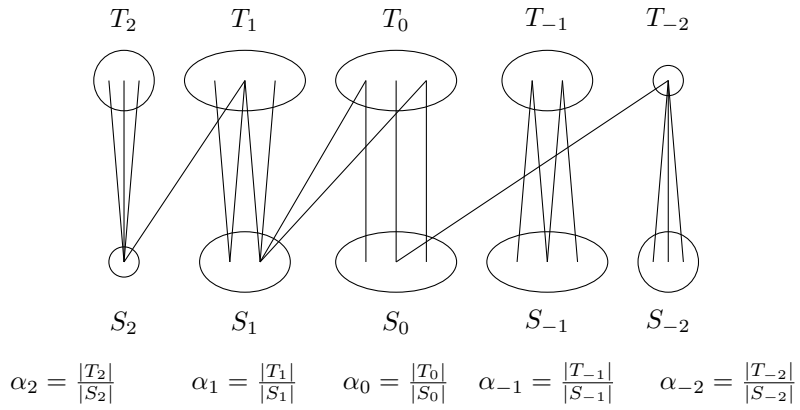


Figure 4.3: Canonical decomposition of a bipartite graph. Note that edges from S_i only go to T_j with $j \leq i$ (property (1)).

the analysis above proceeds by showing that not too much mass will be in the tail $\int_k^\infty b^k(x)dx$, here we find it more convenient to show that substantial mass will be in the head of the distribution, i.e. bound $\int_0^k b^k(x)dx$ from below. We extend the argument using a careful reweighting of vertices and scaling of levels guided by the structure of the *canonical decomposition* of G introduced in [35], which we now define.

Let $G = (P, Q, E)$ denote a bipartite graph. For a set $S \subseteq P$ we denote the set of neighbors of S by $\Gamma(S)$. For a number $\alpha > 0$ the graph G is said to have vertex expansion at least α if $|\Gamma(S)| \geq \alpha|S|$ for all $S \subseteq P$.

Definition 88 (Canonical decomposition) *Let $G = (P, Q, E)$ denote a bipartite graph. A partition of $Q = \bigcup_{j\in\mathcal{I}} T_j, T_j \cap T_i = \emptyset, j \neq i$ and $P = \bigcup_{j\in\mathcal{I}} S_j, S_j \cap S_i = \emptyset, j \neq i$ together with numbers $\alpha_j > 0$, where $\alpha_j \leq 1$ for $j \leq 0$ and $\alpha_j > 1$ for $j > 0$ is called a* canonical partition *if*

1. *for all i one has $\Gamma\left(\bigcup_{j\in\mathcal{I}, j\leq i} S_j\right) \subseteq \bigcup_{j\in\mathcal{I}, j\leq i} T_j$;*
2. *$|\Gamma(S) \cap T_j| \geq \alpha_j|S|$ for all $S \subseteq S_j$ for all $j \in \mathcal{I}$;*
3. *$|T_j|/|S_j| = \alpha_j, j \in \mathcal{I}$.*

Here $\mathcal{I} \subset \mathbb{Z}$ is a set of indices.

Please see Fig. 4.3 for an illustration.

Remark 89 *For $k = 1$, the analysis is inspired by the analysis of the round-robin algorithm in [73]. We note that the difference in our case is that we essentially consider a fractional version of their process, and obtain significantly better bounds on the quality of approximation. In particular, the best approximation factor that follows from the result of [73] is* $1/8$ even after any k passes, *while here we get the optimal $1 - 1/e$ factor for $k = 1$, and an approximation of the form $1 - O(1/k^{1/2})$ for all $k > 0$.*

We now introduce some definitions. For a node $v \in Q$ let $l^k(v)$ denote the load of v after the k-th pass. Note that $l^k(v) \geq 0$ and may in general grow with $|P|$ for the most loaded vertices in Q. The core of our analysis will consist of bounding the distribution of water levels among vertices in Q, showing that there cannot be too many highly overloaded vertices. It will be convenient to assume that water is allocated in multiples of some $\Delta_{org} > 0$ (such a Δ_{org} always exists since we are dealing with a finite process).

Shadow allocation and density function $\phi_v^k(x)$. First, define

(Source capacities) Define $w_s(u), u \in P$ by setting $w_s(u) = \min\{1, \alpha_j\}$ for $u \in S_j$. Note that one has $\sum_{u \in P} w_s(u) = |M|$. Also, for $v \in T_j$ let $w_s(v) := \min\{1, \alpha_j\}$ for convenience.

(Sink capacities) Define $w_t(v), v \in Q$ by setting $w_t(v) = \min\{1, 1/\alpha_j\}$ for $v \in T_j$. Note that one has $\sum_{v \in Q} w_t(v) = |M|$. Also, for $u \in S_j$ let $w_t(u) := \min\{1, 1/\alpha_j\}$ for convenience.

We will use the concept of a *shadow allocation*, in which whenever a units of water are added to a vertex $v \in Q$ in the original allocation, $a/w_t(v)$ units of water are added to v in the shadow allocation. Now whenever water from a vertex $u \in P$ is added to vertex $v \in Q$ at level x during the j-th pass *in the shadow allocation*, we let $\phi_v^j(x) := w_s(u)$, where ϕ is the *density function*. It will be crucial that

$$\sum_{v \in Q} w_t(v) \int_0^\infty \phi_v^j(x) dx = |M| \tag{4.25}$$

for all $j = 1, \ldots, k$. We assume that water in the shadow allocation is allocated in multiples of some $\Delta > 0$.

Then

Lemma 90 *One has for all $x \geq 0$ and all $k \geq 1$*

$$b^k(x) \geq \int_x^\infty \sum_{v \in Q} w_t(v) \phi_v^k(s) ds.$$

Proof: Recall that the pairs in the canonical decomposition of G are denoted by (S_j, T_j), where the expansion factors α_j are increasing with j. We need to that

$$b^k(x) \geq \int_x^\infty \sum_{v \in Q} w_t(v) \cdot \phi_v^k(s) ds \tag{4.26}$$

By definition of the canonical decomposition $(S_j, T_j)_{j \in \mathcal{I}}$ for each $j \in \mathcal{I}$ there exists a (possibly fractional) matching M_j in G that matches each $u \in S_j$ exactly α_j times and each $v \in T_j$ exactly once. Let $M_j(u, v) \in [0, 1]$ denote the extent to which u is matched to v, so that $\sum_{v \in T_j} M_j(u, v) = \alpha_j$ for all $u \in S_j$ and $\sum_{u \in S_j} M_j(u, v) = 1$ for all $v \in T_j$.

Consider a node $u \in S_j$ and suppose that a Δ_{org} amount of its water was allocated to level $[i \cdot \Delta, (i+1) \cdot \Delta]$ of a vertex $v \in T_r$ *in the original allocation*. Note that $r \leq j$ since there are no

edges from S_j to $T_r, r > j$. By the definition of the shadow allocation Δ_{org} amount of water in the original allocation corresponds to $\frac{\Delta_{org}}{w_t(u)}$ water placed contiguously in the shadow allocation.

By definition of the water-filling algorithm all neighbors w of u must have been at level at least $\frac{w_t(v)}{w_t(u)}(i+1)\Delta_{org} \geq (i+1)\Delta_{org}$ when the node u was allocated since $w_t(v) = \min\{1, 1/\alpha_r\} \geq w_t(u) = \min\{1, 1/\alpha_j\}$.

Thus, we have that for each such $u \in S_j$

$$\text{contribution to rhs of (4.26)} = \Delta_{org} \cdot \phi_v^k(s)$$

since the Δ_{org} amount of water corresponds to $t = \frac{\Delta_{org}}{w_t \Delta}$ slabs of size Δ, and the contribution is then weighted by $w_t(v)$ in the rhs of (4.26). We now calculate the contribution of $u \in S_j$ to the lhs. We let each $u_j \in S_j$ contribute $M_j(u, w)$ to each $w \in T_j$, so that the total contribution to each w is 1 and total contribution of each $u \in S_j$ is α_j. Thus,

$$\text{contribution of } u \text{ to lhs of (4.26)} \geq \Delta_{org} \cdot \min\{1, 1/\alpha_j\} \cdot \alpha_j$$

since u has α_j matches in T_j, whose contributions are weighted by $\min\{1, 1/\alpha_j\}$. As before, it remains to note that $\min\{1, 1/\alpha_j\} \cdot \alpha_j = \min\{1, \alpha_j\} = \phi_v^k(s)$. ■

We now get

Lemma 91 *One has for all $x \geq 0$ and all $k \geq 1$*

$$|M| - b^k(x) \leq \int_0^x (b^k(s) - b^{k-1}(s))ds.$$

Proof: By Lemma 90 we have

$$b^k(x) \geq \int_x^\infty \sum_{v \in Q} w_t(v)\phi_v^k(s)ds.$$

Putting this together with (4.25) we get

$$|M| - b^k(x) \leq \int_0^x \sum_{v \in Q} w_t(v)\phi_v^k(s)ds$$

for all $x \geq 0$ and $k \geq 1$. To complete the proof, we note that

$$\int_0^x \sum_{v \in Q} w_t(v)\phi_v^k(s)ds \leq \int_0^x (b^k(s) - b^{k-1}(s))ds$$

for all $k \geq 1$ and $x \geq 0$, where we let $b^0 \equiv 0$ for convenience. ■

We also need

Lemma 92 *Algorithm 1 constructs a matching of size at least*

$$\frac{1}{k}\int_0^k b^k(x)dx.$$

Proof: A vertex $v \in Q$ contributes $\frac{1}{k}\min\{k, l^k(v)\} \geq w_t(v)\frac{1}{k}\min\{k, l^k(v)/w_t(v)\}$ to the matching, implying that the size of the constructed matching is at least

$$\frac{1}{k}\int_0^k b^k(x)dx.$$

■

We now prove lower bounds on $b^k(x)$. Recall that for integer $k \geq 1$

$$F^k(x) = \int_x^\infty e^{-s}s^{k-1}/(k-1)!ds = \sum_{i=0}^{k-1} e^{-x}x^i/i!. \tag{4.27}$$

Note that $1 - F^k(x)$ is the cdf of the Gamma distribution with scale 1 and shape k.

Lemma 93 *For every $k \geq 1$ one has for all $x \geq 0$*

$$\int_0^x b^k(s)ds \geq |M| \cdot \int_0^x F^k(s)ds.$$

Proof:

We prove the lemma by induction on k.

Base: $k = 1$ This follows immediately since by Lemma 91 one has

$$\int_0^x b^1(s)ds \geq |M| - b^1(x). \tag{4.28}$$

Letting $f(x) = \int_0^x b^1(s)ds$, we get that $f'(x) \geq |M| - f(x)$ for all $x \geq 0$, $f(0) = 0$ and $f'(0) = |M|$, which implies that $f(x) \geq |M| \cdot (1 - e^{-x})$, as required.

Inductive step: $k - 1 \to k$ We need to prove that

$$\int_0^x b^k(s)ds \geq |M| \cdot \int_0^x F^k(s)ds. \tag{4.29}$$

By Lemma 91 for all $x \geq 0$

$$b^k(x) \geq |M| - \int_0^x (b^k(s) - b^{k-1}(s))ds \geq |M| - \int_0^x b^k(s)ds + |M| \cdot \int_0^x F^{k-1}(s)ds, \tag{4.30}$$

where we used the inductive hypothesis to replace $\int_0^x b^{k-1}(s)ds$ with $|M| \cdot \int_0^x F^{k-1}(s)ds$.

Thus,

$$\int_0^x b^k(s)ds \geq |M| - b^k(x) + |M| \cdot \int_0^x F^{k-1}(s)ds. \tag{4.31}$$

Let $f(x) = \int_0^x b^k(s)ds$. We have from (4.31) that

$$f'(x) = |M| - f(x) + |M| \cdot \int_0^x F^{k-1}(s)ds, f(0) = 0, f'(0) = |M|.$$

Thus, $f'(x)$ is given by the solution of

$$g(x) = -g'(x) + |M| \cdot F^{k-1}(x), g(0) = |M|. \tag{4.32}$$

The solution of (4.32) is given by

$$g(x) = e^{-x}\left(|M| \int_0^x e^s F^{k-1}(s)ds + |M|\right). \tag{4.33}$$

Calculating the integral in (4.33) yields

$$\int_0^x e^s F^{k-1}(s)ds = \int_0^x e^s \int_s^\infty \frac{1}{(k-1)!} z^{k-1}e^{-z}dzds = \int_0^x \sum_{j=0}^{k-1} \frac{1}{j!}s^j ds = \sum_{j=1}^{k} \frac{1}{j!}x^j, \tag{4.34}$$

and hence $g(x) = |M| \cdot F^k(x)$. Thus, $\int_0^x b^k(s)ds \geq f(x) = |M| \cdot \int_0^x F^k(s)ds$ as required.

■

Given Lemma 93, we immediately obtain

Theorem 94 *Algorithm 1 achieves a* $(1 - e^{-k}\frac{k^{k-1}}{(k-1)!})$*-approximation to maximum matchings in* k *passes over the input stream.*

Proof: The approximation ratio is at least

$$\frac{1}{k}\int_0^k b^k(x)dx \geq \frac{1}{k}\int_0^k F^k(x)dx = 1 - \frac{1}{k}\int_k^\infty F^k(x)dx.$$

Recalling that $F^k(x) = \sum_{j=0}^{k-1} e^{-x}x^j/j!$ and using integration by parts

$$\int e^{-x}x^j/j!dx = -e^{-x}x^j/j!\Big|_k^\infty + \int e^{-x}x^{j-1}/(j-1)!dx,$$

we get

$$\begin{aligned}\int_k^\infty F^k(x)dx &= \int_k^\infty \sum_{j=0}^{k-1} e^{-x}x^j/j!dx = \sum_{j=0}^{k-1}(k-j)e^{-k}k^j/j! \\ &= \sum_{j=0}^{k-1} e^{-k}k^{j+1}/j! - \sum_{j=1}^{k-1} e^{-k}k^j/(j-1)! = e^{-k}k^k/(k-1)!\end{aligned} \tag{4.35}$$

Thus,

$$\frac{1}{k}\int_k^\infty F^k(x)dx = \frac{e^{-k}k^{k-1}}{(k-1)!} = \frac{1}{\sqrt{2\pi k}} + O(k^{-3/2})$$

■

4.4 Gap-existence

In this section we show how our techniques yield an efficient algorithm for Gap-existence, thereby proving Theorem 71. Recall that we are given a graph $G = (A, I, E)$ and integral budgets B_a. Note that integral budgets can be simulated implicitly by creating B_a copies of a for all $a \in A$. For simplicity, this is the approach that we take.

We now present a discretized version of Algorithm 1. We will explicitly maintain a subset $I^* \subset I$ of size $O(|A|/\epsilon)$, relying on the following two oracles:

1. an oracle LIST-NEIGHBORS(a, I^*) that, given a node $a \in A$ and a set I^* outputs the set of nodes $I^{**} \subseteq I^*$ that a is connected to;
2. an oracle NEW-NEIGHBOR(a, I^*) that, given any set $I^* \subseteq I$, outputs any node $i \in I \setminus I^*$ that a is connected to or $\emptyset$ if all neighbors of a are in I^*.

Algorithm 2: DISCRETIZED-WATERFILLING(G, a, ϵ, k)

```
I* ← ∅
while exists a neighbor i of a in I* with level < (ε/4)k do
    Allocate water to i until it is at level (ε/4)k
end while
I* ← I* ∪ NEW-NEIGHBOR(a, I*)
Perform water filling on neighbors in I*.
REMOVE-CYCLES(G')
```

First we prove

Lemma 95 *The space used by Algorithm 2 is $O(|A|/\epsilon)$.*

Proof: Call a vertex *saturated* if the amount of water in it is at least ϵk. The number of saturated vertices is $O(|A|/\epsilon)$ since there are $k|A|$ units of water in the system, and each saturated vertex accounts for at least ϵk. We say that an unsaturated vertex i belongs to $a \in A$ if i was added to I^* when NEW-NEIGHBOR was called from a. Note that for each $a \in A$ only one $i \in I$ belongs to a. Thus, this amounts to at most $|A|$ additional vertices. ■

Our algorithm for Gap-Existence is as follows:

Algorithm 3: GAP-EXISTENCE(G, ϵ)

```
Run DISCRETIZED-WATERFILLING(G) with k = O(log(|I| · Σ_{a∈A} B_a)/ε²).
Let G' denote the support of the fractional solution.
Output YES if a complete matching with budgets ⌊(1 − ε)B_a⌋ exists in G', NO otherwise.
```

We now assume that we are in the **YES** case and prove that the algorithm will find a matching with budgets $\lfloor (1-\epsilon)B_a \rfloor$. We refer to vertices $i \in I$ that have a nonzero amount of water as *active*. Let $p_i = 1$ for active vertices and $p_i = 0$ o.w. Abusing notation somewhat, for an active vertex $i \in I$ let $l^k(i)$ denote the level of water in i minus ϵk and 0 otherwise. The Gap-Existence case is in fact somewhat simpler than the general case of approximating matchings that we just discussed, so we will use the more lightweight techniques from the analysis of the simple case for matchings.

For each $k \geq 1$ and all $x \geq 0$ denote by $b^k(x)$ the number of vertices in I that have load *at least* $x + \epsilon k$ after k passes. We start by pointing out some useful properties of the function $b^k(x)$. First, note that $b^k(0) \leq |I|$, $b^k(x)$ is non-increasing in x and $b^k(x) - b^{k-1}(x) \geq 0$ for all x. Recall that we are interested in recovering a $1-\epsilon/2$-matching of the A side. To do that, we scale all allocations by $1-\epsilon/2$. The size of the matching recovered is

$$(1-\epsilon/2)(\epsilon/4)k \sum_{i \in I} p_i + (1-\epsilon/2) \int_0^\infty b^k(x)dx = (1-\epsilon/2)k|M|, \tag{4.36}$$

since every vertex $a \in A$ contributed k units of water, one in each round, amounting to $k|M|$ amount of water overall, except for the water that was allocated below ϵk, and (4.36) calculates the sum of loads on all $i \in I$. Furthermore, note that the size of the matching constructed by the algorithm after k passes is at least

$$(1-\epsilon/2)(\epsilon/4) \sum_{i \in I} p_i + \frac{1}{k} \int_0^{k(1-\epsilon/4)/(1-\epsilon/2)} b^k(x)dx, \tag{4.37}$$

since every vertex $i \in I$ with load x contributes at least $\frac{1}{k} \cdot \min\{k(1-\epsilon/4), x\}$ to the matching before scaling, and hence $\frac{1}{k} \cdot \min\{k(1-\epsilon/4)/(1-\epsilon/2), x\}$ after scaling. Hence the approximation ratio after k passes is at least

$$1 - \frac{1}{k} \int_{k(1-\epsilon/4)/(1-\epsilon/2)}^\infty b^k(x)dx, \tag{4.38}$$

where we used (4.36) to convert (4.37) into (4.38). Thus, it is sufficient to lower bound $\int_0^k b^k(x)dx$ in order to analyze the approximation ratio, and we turn to bounding this quantity.

Lemma 96 *One has for all $k \geq 1$*

$$b^k(x) \geq \int_x^\infty (b^k(s) - b^{k-1}(s))ds. \tag{4.39}$$

for all $x \geq 0$, where $b^0 \equiv 0$.

Proof: For each such vertex $a \in A$ consider its match $M(a)$. If a ended up allocating water at level at least x during the k-th pass, its match $M(a)$ must have been at level at least x when a arrived. Together with the fact that levels are monotone increasing this gives the result. We omit the details since they would essentially repeat the proof of Lemma 90 with minor changes due to the absence of weights w_t on the I side. ■

We now get

Lemma 97 *For all $k \geq 1$ and all $x \geq 0$*

$$\int_x^\infty b^k(s)ds \leq |I| \cdot \int_x^\infty F^k(s)ds. \tag{4.40}$$

Proof: We prove the lemma by induction on k.

Base: $k = 1$ We prove the statement by contradiction. Suppose that

$$\int_{x_0}^\infty b^1(s)ds > |I| \int_{x_0}^\infty e^{-s}ds = |I|e^{-x_0} \tag{4.41}$$

for some $x_0 \geq 0$. Recall that by Lemma 96 one has

$$b^1(x) \geq \int_x^\infty b^1(s)ds, \tag{4.42}$$

for all $x \geq 0$. Let $g(x) = \left(\int_{x_0}^\infty b^1(s)ds\right) \cdot e^{-x+x_0}$ for $x \in [0, x_0]$. Then $g(x)$ satisfies (4.42) with equality, and hence $b^1(x) \geq g(x)$ for all $x \in [0, x_0]$. But $g(0) = \left(\int_{x_0}^\infty b^1(s)ds\right) \cdot e^{x_0} > |I|$, a contradiction with $b^1(0) = |I|$.

Inductive step: $k - 1 \to k$ We need to prove that

$$\int_x^\infty b^k(s)ds \leq |I| \cdot \int_x^\infty F^k(s)ds. \tag{4.43}$$

Recall that by Lemma 96 for all $x \geq 0$

$$b^k(x) \geq \int_x^\infty (b^k(s) - b^{k-1}(s))ds = \int_x^\infty b^k(s)ds - |I| \cdot \int_x^\infty F^{k-1}(s)ds, \tag{4.44}$$

where we used the inductive hypothesis to replace $\int_x^\infty b^{k-1}(s)ds$ with $|I| \cdot \int_x^\infty F^{k-1}(s)ds$.

Fix any point $x_0 \geq 0$ and denote

$$\gamma := \int_{x_0}^\infty b^k(s)ds.$$

We will show that one necessarily has $\gamma > |I| \cdot \int_{x_0}^\infty F^k(s)ds$.

It now follows from (4.44) that $b^k(x)$ is lower bounded by the solution of

$$g(x) = \int_x^\infty (g(s) - |I| \cdot F^{k-1}(s))ds$$
$$g(x_0) = \gamma.$$

Thus, $g(x)$ satisfies

$$g'(x) = -g(x) + |I| \cdot F^{k-1}(x), g(x_0) = \gamma. \tag{4.45}$$

The solution of (4.45) is given by

$$g(x) = e^{-x}\left(-|I|\int_0^x e^s F^{k-1}(s)ds + c\right), \tag{4.46}$$

where the constant c depends on γ. Note that $g(0) = c$, and recalling that g lower bounds $b^k(x)$, which is at most $|I|$ at $x = 0$, we have that $c \leq |I|$.

Calculating the integral in (4.46) yields

$$\int_0^x e^s F^{k-1}(s)ds = \int_0^x e^s \int_s^\infty \frac{1}{(k-1)!} z^{k-1}e^{-z}dzds = \int_0^x \sum_{j=0}^{k-1} \frac{1}{j!}s^j ds = \sum_{j=1}^{k} \frac{1}{j!}x^j, \tag{4.47}$$

and hence

$$g(x) = (c + |I|\sum_{j=1}^{k} e^{-x}x^j/j!) = |I| \cdot F^k(x) + (c - |I|).$$

In particular, it follows that $\gamma = g(x_0) = |I| \cdot F^k(x) + (c - |I|) \leq |I| \cdot F^k(x)$, completing the proof of the inductive step.

■

We now ready to prove correctness. Suppose that we are in the **YES** case, i.e. there exists a complete matching with budgets B_a. Consider the fractional allocation returned by DISCRETIZED-WATERFILLING(G), and multiply it by $(1-\epsilon/2)/k$. Recalling that each active vertex can take at least $1-\epsilon/4$ units of water, we get that every vertex in $i \in I$ now contributes $\frac{1-\epsilon/2}{k(1-\epsilon/4)} \min\{k(1-\epsilon/4)/(1-\epsilon/2), l^k(v)\}$ to the matching.

Thus, by Lemma 97 together with (4.38) shows that the amount of water lost is at most

$$(1-\epsilon/2)|I|\frac{1}{k}\int_{k(1-\epsilon/4)/(1-\epsilon/2)}^{\infty} b^k(x)dx. \tag{4.48}$$

We will need

Lemma 98 *For all $k \geq 1$ and $\epsilon^* \geq 0$*

$$\frac{1}{k}\int_{k(1+\epsilon^*)}^{\infty} F^k(x)dx \leq e^{-\epsilon^* k}(1+\epsilon^*)^k \cdot e^{-k}k^{k-1}/(k-1)!$$

Proof: Recalling that $F^k(x) = \sum_{j=0}^{k-1} e^{-x}x^j/j!$ and using integration by parts

$$\int e^{-x}x^j/j!dx = \left[-e^{-x}x^j/j!\right]_k^\infty + \int e^{-x}x^{j-1}/(j-1)!dx,$$

we get

$$\begin{aligned}\int_{k(1+\epsilon^*)}^{\infty} F^k(x)dx &= \int_{k(1+\epsilon^*)}^{\infty} \sum_{j=0}^{k-1} e^{-x}x^j/j!dx = \sum_{j=0}^{k-1}(k-j)e^{-k(1+\epsilon^*)}(k(1+\epsilon^*))^j/j! \\ &\leq e^{-\epsilon^* k}(1+\epsilon^*)^k \sum_{j=0}^{k-1}(k-j)e^{-k}k^j/j! = e^{-\epsilon^* k}(1+\epsilon^*)^k \cdot e^{-k}k^{k-1}/(k-1)!\end{aligned} \tag{4.49}$$

■

Let $\epsilon^* = (1-\epsilon/4)/(1-\epsilon/2) - 1$. By Lemma 98 we have

$$\frac{1}{k}\int_{k(1+\epsilon^*)}^{\infty} b^k(x)dx \leq e^{-k(\epsilon^* - \ln(1+\epsilon^*))} \cdot [e^{-k}k^{k-1}/(k-1)!] \leq e^{-(\epsilon^*)^2 k/3} \tag{4.50}$$

for sufficiently small $\epsilon^* > 0$. Also note that $\epsilon/5 \leq \epsilon^* \leq \epsilon$ for sufficiently small $\epsilon > 0$. Hence, letting $k = \gamma \log(|I| \cdot \sum_{a\in A} B_a)/\epsilon^2$ for a sufficiently large constant $\gamma > 0$ yields a $(1 - (\sum_{a\in A} B_a)^{-2})$-approximate fractional matching with budgets $\lfloor(1-\epsilon)B_a\rfloor$. We now argue that the set of edges that this fractional matching is supported on admits a complete matching with budgets $\lfloor(1-\epsilon)B_a\rfloor$. We will need

Lemma 99 *Let $G = (P, Q, E)$ denote a bipartite graph. Suppose that there exists a fractional matching of size $|P|(1-|P|^{-2})$ in G. Then the support of the fractional matching contains a perfect matching of the $|P|$ side.*

Proof: Consider the subgraph G' that supports a fractional $1 - |P|^{-2}$ matching. Recall that a graph supports an α-matching of the P-side iff $|\Gamma(S)| \geq \alpha|S|$ for all $S \subseteq P$. Now note that the ratio $|\Gamma(S)|/|S|$ is a rational number of the form i/j where $j \leq |P|$. The existence of the fractional matching implies that $|\Gamma(S)|/|S| \geq (1 - |P|^{-2})$ for all $S \subseteq P$. Since $|\Gamma(S)|/|S|$ can only have denominator at most $|P|$, this implies that in fact $|\Gamma(S)| \geq |S|$ for all $|S|$. ■

Since the budgets $\lfloor(1-\epsilon)B_a\rfloor$ are integral, finding a complete matching with budgets $\lfloor(1-\epsilon)B_a\rfloor$ is equivalent to finding a complete matching in a graph with $\sum_{a\in A}\lfloor(1-\epsilon)B_a\rfloor$ vertices on the A side. Lemma 99 now implies the existence of a complete matching in the set of edges that the fractional matching is supported on. This completes the proof of Theorem 71.

Chapter 5

Spectral sparsification via random spanners

Our results in Chapters 2-4 revolved around matching problems. We showed in Chapter 2 how cut-preserving sparsification turns out to be useful for finding perfect matchings in *regular* bipartite graphs in sublinear time, and developed a more general notion of sparsification relevant to the problem of finding matchings in the streaming model in Chapter 3. In this chapter we study the relationship between two different notions of sparsification, namely *spanners*, i.e. subgraphs that preserve distances approximately, and *spectral* sparsification, which is a generalization of cut-preserving sparsification. We uncover a connection between these two notions of sparsification by defining a new notion of distance between nodes of the graph, which we call *robust connectivity*, and relating it to *effective resistance* between nodes in the graph. We also show how to approximate such distances efficiently in small space, i.e. obtain an *oracle* for robust connectivity. This oracle allows us to obtain a nearly-linear time algorithm for constructing spectral sparsifiers.

Before stating our results on robust connectivity we discuss the relevant notions distances that have been studied in the literature. Various notions of distance between nodes of the graph have been considered recently, e.g. shortest path distance, minimum cuts, effective resistance etc. For all of these notions of 'distance' it is known how to compress a graph to a small representation that allows to compute 'distance' queries approximately from the compressed representation. For example, for shortest path distance this is provided by the spanner construction algorithms of Thorup and Zwick[89] (see also [90, 70, 8, 9, 10, 11]), for cuts this is given by sparsifiers of Benczúr and Karger[16] (see also [30]) and for effective resistances by spectral sparsifiers of Spielman and Srivastava [83] (see also [12]). In fact, all of these methods for succint representation satisfy a stronger guarantee – they support efficient queries of the corresponding 'distances' between nodes. It should be noted here that the latter two measures of distance essentially take into account *the set of paths* between a pair of nodes in the graph as opposed to the path of minimum length.

Here we introduce a new notion of distance between nodes in a graph that we refer to as *robust connectivity*. Robust connectivity between a pair of nodes u and v is parameterized by a threshold κ and intuitively captures the number of paths between u and v of length at most κ. Using this new notion of distances, we show that any black box algorithm to construct a spanner can be viewed as an algorithm to construct a sparsifier: Given a graph G, simply take the spanners of a few (polylogarithmically many) random subgraphs of G obtained by sampling edges at different probabilities; the union of these spanners, appropriately weighted, is also a sparsifier of G. While the cut sparsifiers of Benczur and Karger were based on weighting edges according to (inverse) strong connectivity, and the spectral sparsifiers were based on resistance, our method weights edges using the robust connectivity measure. The main property that we use is that this new measure is always greater than the resistance when scaled by a factor of $O(\kappa)$, but (just like resistance and connectivity) – has a bounded sum ($\tilde{O}(n)$) over all the edges of the graph. Our distance measure can be viewed as a generlization of the affinity measure that was used in an experimental paper to study user behavior in social networks [76].

We now give an outline of our results. In order to simplify presentation, we now define robust connectivities for unweighted graphs. The definition for weighted graphs will be given in Section 5.2. We stress here that even though we give the definition for unweighted graphs now, we will prove

all our results for weighted graphs. Let $G = (V, E)$ be an undirected graph. For a sampling probability $p \in (0, 1)$ denote by G_p the graph obtained by sampling edges of G with probability p. Let $d_G(u, v)$ denote the shortest path distance between u and v in G. For a pair of nodes, the *κ-Robust Connectivity* is the highest sampling probability at which the they are at least distance κ apart in G_p with constant probability.

Definition 100 (Robust Connectivity) *For a pair of nodes (u, v) let the κ-robust connectivity $q_\kappa(u, v)$ denote the largest $\eta \in (0, 1]$ such that $Pr[d_{G_\eta}(u, v) \geq \kappa] \geq 1/2$. For an edge $e = (u, v)$, we use $q_\kappa(e)$ to denote $q_\kappa(u, v)$.*

We show that the robust connectivity upper bounds the resistance (upto a $O(\kappa)$ factor) and also has a bounded sum over all edges.

Lemma 101 *For all $e \in E$*

$$q_\kappa(e) \geq R_e/(2\kappa).$$

and

$$\sum_{e \in E} q_\kappa(e) \leq 2n^{1+O(1/\kappa)}$$

Further, based on distance oracles, which support approximate shortest path distance queries efficiently, one can construct an oracle that supports queries of approximate - robust connectivities between any pair of nodes.

Lemma 102 *For a fixed graph G, there is a oracle that for any pair of nodes (u, v), can be used to query an estimate $\hat{q}_\kappa(u, v)$ of κ-robust connectivity. The estimate satisfies the (slightly weaker) conditions: for all $e \in E$*

$$\hat{q}_\kappa(e) \geq R_e/(2\kappa^2).$$

and

$$\sum_{e \in E} \hat{q}_\kappa(e) \leq O(n^{1+O(1/\kappa)})$$

If $\kappa = \Omega(\log n)$, the oracle can be constructed in time $\tilde{O}(m)$, stores a sketch of size $\tilde{O}(1)$ per node, and each query takes $\tilde{O}(1)$ time.

We note that Lemma 102 immediately yields a simple $\tilde{O}(m)$ time algorithm for spectral sparsification.

Let S be any black box algorithm spanner construction algorithm, that on input G outputs a spanner $S(G)$. Assume that the distances between all pairs of nodes in $S(G)$ are within factor $O(\log n)$ of the true distance in G. Existing spanner construction algorithms produce such spanners with $O(n)$ edges in time $\tilde{O}(m)$. We show

Theorem 103 *Let $\{G_{i,j} : 1 \leq i \leq O\left(\log_{\frac{1}{1-\epsilon}}\left(\frac{n^4 w_{max}}{w_{min}}\right)\right),\ 1 \leq j \leq O(\log^3 n/\epsilon^3)\}$ be a collection of random subgraphs of G, where $G_{i,j}$ is an independent copy of G_p for $p = \frac{1}{w_{min}}(1-\epsilon)^i$. Then there is a weighting of the edges of the subgraph $H = \cup_{i,j} S(G_{i,j})$ such that it is a $(1 \pm \epsilon)$-sparsifier of G. Moreover, such a weighting can be constructed in $\tilde{O}(m)$ time.*

Organization We start by introducing definitions related to spectral sparsification and spanners in section 5.1. In section 5.2 we give the definition of robust connectivities and prove Lemma 101 and Lemma 102, thus obtaining a simple algorithm for spectral sparsification. Finally, in section 5.3 we prove Theorem 103.

5.1 Background and notation

For a weighted undirected graph $G = (V, E, w)$ the Laplacian matrix of G is the matrix defined as

$$L_G(i,j) = -w_{(i,j)} \text{ and } L_G(i,i) = \sum_{j \neq i} w_{ij}$$

Definition 104 (Spectral ordering of graphs) *We define a partial ordering $\prec$ on graphs by letting*

$$G \prec H \text{ if and only if } x^T L_G x \leq x^T L_H x \ \forall x \in \mathbb{R}^{|V|}.$$

A weighted undirected graph G can be associated with an electrical network with link e having conductance w_e (i.e. corresponding to a resistor of resistance $1/w_e$). Then the effective resistance R_e across an edge e is the potential difference induced across it when a unit current is injected at one end of e and extracted at the other end of e. We will use the following

Theorem 105 (Spectral sparsification, [83]) *Let H be obtained by sampling edges of G independently with probability $p_e = \Theta(w_e R_e \log n/\epsilon^2)$ for some $\epsilon > 1/\sqrt{n}$ and giving each sampled edge weight $1/p_e$. Then whp*

$$(1-\epsilon)G \prec H \prec (1+\epsilon)G.$$

The following corollary is well-known (see, e.g.[64]):

Corollary 106 (Oversampling) *Let H be obtained by sampling edges of G independently with probability $p_e \geq c w_e R_e \log n/\epsilon^2$ for some $\epsilon > 1/\sqrt{n}$ and a sufficiently large constant $c > 0$, and giving each sampled edge weight $1/p_e$. Then whp*

$$(1-\epsilon)G \prec H \prec (1+\epsilon)G.$$

We will also need definitions related to spanners.

Definition 107 (t-spanner) *A t-spanner of a weighted undirected graph G is a subgraph H of G such that the distances in H are stretch t estimates of the distances in G, i.e. for all $u, v \in V$ one has*

$$d_G(u,v) \leq d_H(u,v) \leq t \cdot d_G(u,v).$$

5.2 Robust connectivities

In this section we define robust connectivities, prove their main properties and give an efficient algorithm for approximating them. These results together with Corollary 106 imply a simple algorithm for spectral sparsification.

Let $G = (V, E)$ be a weighted undirected graph with edge weights w_e. For a sampling parameter $p \in \mathbb{R}^+$ denote by G_p the *unweighted* random graph obtained by sampling each edge $e \in E$ independently with probability $\min\{w_e p, 1\}$. For a graph H we denote the shortest path distance between nodes u and v in H by $d_H(u, v)$. We start by defining *robust connectivities*, for which we need an auxiliary definition.

Definition 108 *Let $G = (V, E)$ be a weighted undirected graph. For $\eta \in \mathbb{R}^+$ and $e \in E$ let $p_\kappa(e, \eta) = \mathbf{Pr}[d_{G_\eta}(u, v) > \kappa]$.*

Note that when G is an unweighted graph, it is sufficient to consider $\eta \in (0, 1]$.

Definition 109 *For $e = (u, v) \in E$ let the κ-robust connectivity $q_\kappa(e)$ denote the largest $\eta \in \mathbb{R}$ such that $p_\kappa(e, \eta) \geq 1/2$.*

In what follows the parameter κ will be fixed, and we will use the term robust connectivity for clarity. We will now show that $q_\kappa(e)$ upper bounds effective resistance up to a factor of κ. We will need

Lemma 110 (Rayleigh's monotonicity principle, [18]) *Cutting an edge of an electrical resistor network does not decrease the effective resistance between any pair of nodes.*

Lemma 111 *For all $e \in E$*

$$q_\kappa(e) \geq R_e/(2\kappa).$$

Proof: We use the characterization of conductance between a pair of nodes in an electrical resistor network as

$$C_{uv} = \min_{x \in \mathbb{R}^V : x_s = 1, x_t = 0} \sum_{e=(u,v) \in E} w_e (x_u - x_v)^2. \tag{5.1}$$

For a sampling parameter $p \in \mathbb{R}^+$ let E_p denote a random set of edges obtained by sampling E independently with probability $\min\{w_e p, 1\}$. Similarly, denote the conductance of $e = (u, v)$ in G_p by

$$C_{uv}(p) = \min_{x \in \mathbb{R}^V : x_s = 1, x_t = 0} \sum_{e=(u,v) \in E_p} (x_u - x_v)^2. \tag{5.2}$$

We have

$$\begin{aligned}\mathbf{E}\left[C_{uv}(p)\right] &= \mathbf{E}\left[\min_{x\in\mathbb{R}^V : x_s=1, x_t=0} \sum_{e=(u,v)\in E_p} (x_u - x_v)^2\right] \\ &\leq \min_{x\in\mathbb{R}^V : x_s=1, x_t=0} \mathbf{E}\left[\sum_{e=(u,v)\in E_p} (x_u - x_v)^2\right] \\ &= \min_{x\in\mathbb{R}^V : x_s=1, x_t=0} \sum_{e=(u,v)\in E} (w_e p)(x_u - x_v)^2 = pC_{uv}\end{aligned}$$

and hence by Markov's inequality, one has

$$\mathbf{Pr}[C_{uv}(p) > 2p/R_e] \leq \mathbf{Pr}[C_{uv}(p) > 2\mathbf{E}[C_{uv}]] \leq 1/2. \tag{5.3}$$

On the other hand, for $p \geq q_\kappa(e)$ one has $\mathbf{Pr}[d_{G_p}(u,v) \leq \kappa] \geq 1/2$. This implies by Lemma 110 that

$$\mathbf{Pr}[C_{uv}(p) \geq 1/\kappa] \geq 1/2 \tag{5.4}$$

since the conductance of a path of length at most κ is at least $1/\kappa$. Setting $p = q_\kappa(e)$ and combining (5.3) and (5.4), we get that

$$2q_\kappa(e)/R_e \geq 1/\kappa, \tag{5.5}$$

i.e. $q_\kappa(e) \geq R_e/(2\kappa)$. ■

We will need the following simple lemma

Lemma 112 *For all $r' < r$ one has $p_\kappa(e,r) \leq p_\kappa(e,r')$.*

Proof: The result follows by coupling the process of sampling edges at rate r' and r such that the set of edges picked by the first process is a subset of edges picked by the second process. ■

We now prove that the sum of distance thresholds over edges of G is small. In order to do that, we relate the distance threshold of $e \in E$ to the probability of including e in a randomized spanner construction which we now define. Note that this construction is only used to prove an upper bound on the sum of distance thresholds and, in particular, we do not provide an efficient implementation here.

The intuition behind the randomized construction of the spanner is very simple, and we outline it here for the case of unweighted graphs. Arrange edges of G in a random order, add for each edge e in that order add e to the spanner H if e does not form a cycle of length smaller than κ with the edges that are included so far. It is known that such a process produces a spanner [7] with at most $n^{1+O(1/\kappa)}$ edges. On the other hand, the set of edges that have rank at most t in a uniformly random order essentially form a random sample of edges of G where each edge is present with probability t/m. Hence, the probability that the endpoints of e are at least κ apart in the subgraph formed by edges with rank at most t, and hence in the spanner constructed so far, is approximately $p_\kappa(e, t/m)$, allowing us to conclude that each edge is taken with probability about $q_\kappa(e)$, since it is taken with

constant probability if it arrives before time $q_\kappa(e) \cdot m$. We now make a version of this argument formal. Since ordering edges by a uniformly random permutation causes dependencies, we choose a slightly different but essentially equivalent approach. We first define the following

Algorithm 4: Randomized spanner construction

```
H ← (V, ∅)
for η = 0 to 1/w_min with step Δ do
  for each e = (u, v) ∈ E do
    γ ← min{1, w_e Δ / (1 − w_e η)}.
    X_e(η) ← Bernoulli(γ).
    if X_e(η) = 1 then
      Add e to H if d_H(u, v) < κ.
    end if
  end for
end for
```

Note that intuitively, the sampling parameter η iterates over the relevant range of sampling parameters $[0, 1/w_{min}]$ with a sufficiently small step Δ, sampling the edges of G in such a way that at each time η the distribution of sampled edges is the same as in G_η. For sufficiently small Δ, we have that $\sum_e X_e(\eta) \leq 1$ for all η with high probability. More precisely, we say that an edge $e \in E$ is *sampled by time* η by Algorithm 4 if $X_e(\eta') = 1$ for at least one $\eta' \in [0, \eta]$. Note that an edge e that is sampled is not necessarily included in H. Denote the set of edges sampled by Algorithm 4 by time η by $E_s(\eta)$. Denote the set of edges included in H by time η by $E_H(\eta)$. The crucial property of Algorithm 4 that we will use in our analysis is

Lemma 113 *$E_s(\eta)$ is distributed as a uniformly random independent sample of edges of E, where edge $e \in E$ is picked with probability* $\min\{1, w_e\eta\}$.

Proof: Since the coin tosses for different edges are independent, it is sufficient to consider a fixed $e \in E$. We prove by induction on η (recall that η iterates over a discrete set) that

$$\mathbf{Pr}[e \in E_s(\eta)] = \min\{1, w_e\eta\}.$$

Base:$\eta = 0$ One has $\mathbf{Pr}[e \in E_s(\eta)] = \min\{1, w_e\eta\} = 0$.

Inductive step:$\eta \to \eta + \Delta$ One has

$$\begin{aligned}
\mathbf{Pr}[e \in E_s(\eta + \Delta)] &= \mathbf{Pr}[e \in E_s(\eta)] \\
+\mathbf{Pr}[e \text{ sampled at step } \eta + \Delta | e \notin E_s(\eta)]&\mathbf{Pr}[e \notin E_s(\eta)] \\
&= \mathbf{Pr}[e \in E_s(\eta)] \\
+\mathbf{Pr}[e \text{ sampled at step } \eta + \Delta]&\mathbf{Pr}[e \notin E_s(\eta)] \\
&= w_e\eta + \min\left\{1, \frac{w_e\Delta}{1 - w_e\eta}\right\}(1 - w_e\eta).
\end{aligned}$$

Since

$$\min\left\{1, \frac{w_e\Delta}{1-w_e\eta}\right\} = \begin{cases} 1, & w_e(\eta+\Delta) \geq 1 \\ \frac{w_e\Delta}{1-w_e\eta} & \text{o.w.} \end{cases},$$

we get that

$$\mathbf{Pr}[e \in E_s(\eta+\Delta)] = \min\left\{1, w_e(\eta+\Delta)\right\}.$$

■

We now (almost) mirror the definition of $q_\kappa(e)$ for the randomized spanner construction for use in the analysis

Definition 114 *Let $q_e^*(\kappa)$ denote the probability of including edge $e \in E$ in H in Algorithm 4, i.e.*

$$q_\kappa^*(e) = \mathbf{Pr}[e \in E_H(R)].$$

We can now prove

Lemma 115 *For all $e \in E$*

$$q_\kappa(e) \leq 2q_\kappa^*(e).$$

Proof: Fix an edge $e \in E$ and let $\eta^* = q_\kappa(e)$. We first note that by Lemma 112, the probability that e is added to H if e is sampled at time η' is only larger than the probability that e is added if it appears at time $\eta > \eta'$. Thus, we have

$$q_\kappa^*(e) \geq \mathbf{Pr}[e \in E_H(\eta^*)] \geq \mathbf{Pr}[e \in E_s(\eta^*)]/2 = q_\kappa(e)/2. \tag{5.6}$$

■

Hence, we get

Lemma 116

$$\sum_{e\in E} w_e q_\kappa(e) \leq 2n^{1+O(1/\kappa)}$$

Proof: One has $\sum_{e\in E} w_e q^*(e) = \mathbf{E}[|E(H)|]$, where H is the random spanner constructed by Algorithm 4. By construction H does not have cycles of length less than κ, and hence $|E(H)| \leq n^{1+O(1/\kappa)}$ (see, e.g. [18]). Now the result follows by Lemma 115. ■

Now the proof of Lemma 101 follows by putting together Lemma 115 and Lemma 111. We now proceed to give an algorithm for efficiently obtaining estimates $\hat{q}_\kappa(u,v)$ of $q_\kappa(u,v)$ guaranteed by Lemma 102.

5.2.1 Estimating $q_\kappa(e)$

We now show how obtain surrogate values $\hat{q}_\kappa(e)$ that we will use in place of $q_e(\kappa)$ such that $\hat{q}_\kappa(e) \geq R_e/\kappa^2$ and $\sum_{e\in E} \hat{q}_\kappa(e) \leq n^{1+O(1/\kappa)}$ in $\tilde{O}(m)$ time. It will be convenient to introduce another

parameter $\delta \in (0,1)$ and write $\hat{q}_{\kappa,\delta}(e)$. Intuitively, δ is the probability that the endpoints of e are more than κ apart a random sample of edges of G where each edge is present independently with probability $\hat{q}_{\kappa,\delta}(e)$, so that $\hat{q}_{\kappa,1/2}(e)$ is an estimate for $q_\kappa(e)$.

The estimation procedure is quite simple. We consider samples of edges of G at a geometric sequence of rates, an calculate the distance between the endpoints of each edge $e \in E$ in the random subgraph given by the sample. Independent repetition of such experiments allows us to estimate the sampling threshold for which the distance between the endpoints of e becomes large. Since the distance calculation is not exact, the approximation to effective resistance that the procedure gives suffers an extra κ factor (we will set $\kappa = \log n$ later). The bound on the sum of the estimated connectivities will follow similarly to the proofs above. We now give a formal description of the estimation procedure.

For each $t = 1, \ldots, T$, where $T = \log(n^4 w_{max}/w_{min})$, and $j = 1, \ldots, J$ for $J = 80 \log n/\delta^2$ define sets E_t^j as follows. For each $e \in E$ add e to E_t^j for t between 1 and $1 + \log(w_e/w_{min})$. Then for each e, repeatedly add e to sets E_t^j with a higher value of t while a coin with heads probability of $1 - \epsilon$ comes up heads.

We will use the Thorup-Zwick distance oracles:

Theorem 117 *[89] Let $G = (V, E)$ be an undirected weighted graph with n vertices and m edges. For any integer $k \geq 1$ the graph G can be preprocessed in $O(kmn^{1/k})$ expected time constructing a data structure of size $O(kn^{1+1/k})$ such that any subsequent distance query can be answered approximately in $O(k)$ time. The approximate distance returned is is of stretch at most $2k - 1$.*

For $t \in [1 : T], j \in [1 : J]$ and $e \in E$ the estimation algorithm sets $\eta_e^j(t) = 0$ if the distance between the endpoints of e is reported by an appropriate distance oracle on E_t^j to be at most κ^2

and 0 otherwise. The formal description is given in Algorithm 5.

Algorithm 5: $ESTIMATE(G, \kappa, \delta)$

for $j = 1$ to J **do**
  Set $E_t^j \leftarrow \emptyset$ for $t \in [1 : T]$.
  For each $e \in E$ add e to E_t^j for t between 1 and $1 + \log(w_e/w_{min})$.
  for $t = 1$ to $T - 1$ **do**
    Add each $e \in E_t^j$ to E_{t+1}^j independently with probability $1 - \epsilon$.
  end for
end for
for $t = 1$ to T **do**
  Construct a Thorup-Zwick distance oracle for E_t^j, $j \in [1 : J]$, denoted by O_t^j.
  for $e = (u, v) \in E$ **do**
    if $O_t^j(u, v) > \kappa^2$ **then**
      $\eta_e^j(t) \leftarrow 1$
    else
      $\eta_e^j(t) \leftarrow 0$
    end if
  end for
end for
for $e \in E$ **do**
  $\hat{q}_{\kappa,\delta}(e) \leftarrow 2^{-t}$, where t is the smallest such that $|\{j : \eta_e^j(t) = 1\}| \geq (1 - \delta)J$
end for
return $\hat{q}_{\kappa,\delta}$

The following lemma gives bounds on $\hat{q}_{\kappa,\delta}$ that yield Lemma 102 after setting $\delta = 1/2$. We will need the ability to chose general δ in the next section (where we will also need property 3 from the lemma below).

Lemma 118 *With high probability for all $e = (u, v) \in E$ one has*

1. $\hat{q}_{\kappa,\delta}(e) \geq \delta R_e/(4\kappa^2)$
2. $\sum_{e \in E} w_e \hat{q}_{\kappa,\delta}(e) \leq 2n^{1+O(1/\kappa)}$
3. *for all $\eta < q_{\kappa,\delta}(e)$ one has* $\mathbf{Pr}[d_{G_\eta}(u, v) > \kappa] \geq 1 - \delta/2$.

Proof: First note that by the choice of $T = \log\left(\frac{n^4 w_{max}}{w_{min}}\right)$ we have for each $e \in E$ that $\mathbf{Pr}[d_{G_{2^{-T}}}(u, v) \geq \kappa] = 1 - n^{-c}$ for a constant $c > 0$, so whp $\hat{q}_{\kappa,\delta}(e)$ is defined by Algorithm 5.

To prove the first statement, we argue similarly to Lemma 111. We use the characterization

$$C_{uv} = \min_{x \in \mathbb{R}^V : x_s = 1, x_t = 0} \sum_{e=(u,v) \in E} w_e (x_u - x_v)^2 \tag{5.7}$$

and denote the conductance of $e = (u, v)$ in G_p by

$$C_{uv}(p) = \min_{x \in \mathbb{R}^V : x_s = 1, x_t = 0} \sum_{e=(u,v) \in E_p} (x_u - x_v)^2. \tag{5.8}$$

As before, we have

$$\mathbf{Pr}[C_{uv}(p) > 2p/(\delta R_e)] \leq \mathbf{Pr}[C_{uv}(p) > (2/\delta)\mathbf{E}[C_{uv}]] \leq \delta/2. \tag{5.9}$$

Let $t^* \in [1, T]$ be such that $\hat{q}_{\kappa,\delta}(e) = 2^{-t^*}$. Let

$$\alpha := \mathbf{Pr}[d_{2\hat{q}_{\kappa,\delta}(e)}(u, v) \geq 1/\kappa^2]$$

It follows by an application of Chernoff bounds that

$$\begin{aligned} \mathbf{Pr}[|\{j : \eta_e^j(t^* - 1) = 1\}| \notin [\alpha - \delta/2, \alpha + \delta/2]J] \\ < e^{-\delta^2 J/16} = n^{-5}. \end{aligned}$$

Hence, we have with high probability that $\alpha \leq 1 - \delta/2$, i.e.

$$\mathbf{Pr}\left[d_{2\hat{q}_{\kappa,\delta}(e)}(u, v) \leq \kappa^2\right] \geq \delta/2. \tag{5.10}$$

Using Rayleigh's monotonicity principle and the fact that the conductance of a path of length at most κ^2 is at least $1/\kappa^2$, we get, combining (5.10) with (5.9)

$$(2/\delta)(2\hat{q}_{\kappa,\delta}(e))/R_e \geq 1/\kappa^2, \tag{5.11}$$

i.e. $q_{\kappa,\delta}(e) \geq \delta R_e/(4\kappa^2)$.

To prove the second inequality, we argue similarly to Lemma 115. Consider a randomized spanner construction process in Algorithm 4 in which an edge e is added to the spanner if there is no path of length smaller than κ in the spanner constructed so far. Since $O_t^j(u, v) \geq \kappa^2$, we have, using the assumption that O_j^i is a κ-approximate distance oracle, that the distance between u and v in the appropriate sampled graph is larger than κ. Thus, similarly to Lemma 115, we have that the probability of including an edge e in the randomized spanner construction is at least $\hat{q}_\kappa(e)/2$. Since the spanner construction process terminates with a graph without cycles of length smaller than κ, we have $\sum_{e \in E} w_e \hat{q}_\kappa(e) \leq 2n^{1+O(1/\kappa)}$. The third inequality follows from the fact the O_t^j is a κ-approximate distance oracle and Chernoff bounds as above. ■

Algorithm 5 immediately yields a simple algorithm for spectral sparsification:

Algorithm 6: Spectral sparsification via robust connectivities

1: Set $\hat{q} \leftarrow ESTIMATE(G, \log n, 1/2) \in E$.

2: Sample each $e \in E$ independently with probability $r_e := \min\{1, (cw_e\hat{q}(e)\log^3 n)/\epsilon^2\}$ for a constant c, giving e weight $1/r_e$ if sampled.

We now obtain

Theorem 119 *Algorithm 4 produces a spectral sparsifier of G with $O(n \log^3 n/\epsilon^2)$ edges whp.*

Proof: It follows from Lemma 118 that $\hat{q}_{\kappa,1/2}(e)(2\log^2 n) \geq R_e$, where R_e is the effective resistance of e. Hence, the sampled graph is a spectral sparsifier whp by Corollary 106. By Lemma 118

$$\sum_{e \in E} w_e \hat{q}_{\kappa,1/2}(\log n) = O(n^{1+O(1/\log n)}) = O(n).$$

Hence, the size of the sample is bounded by $O(n \log^3 n/\epsilon^2)$. ■

In the next section we show that in fact a spectral sparsifier can obtained by taking a union of black-box spanners of random subgraphs of G, thus proving Theorem 103.

5.3 Sparsification by spanners

In this section we give an algorithm for obtaining a spectral sparsifier of an undirected weighted graph G using black box invocations of a spanner construction algorithm, proving Theorem 103. The main challenge here is to overcome dependencies in the sampling process that arise from using a black-box spanner construction. In order to do that, we introduce a procedure that, given a vector of target sampling probabilities $q(e) := \hat{q}_{\kappa,\epsilon}(e), e \in E$, samples edges of G using the black-box spanner construction such that the sampling process (a) stochastically dominates the process of sampling edges independently with probabilities $(1-\epsilon)\min\{1, w_e q(e)\}$ and (b) is stochastically dominated by the process that samples edges independently with probability $\min\{1, w_e q(e)\}$. The procedure consists of a logarithmic number invocations of a basic sampling scheme that we now define.

We first define a sequence of sets $E_1, E_2, \ldots, E_H$ with $H = \log_{1/(1-\epsilon)}\left(\frac{n^4 w_{max}}{w_{min}}\right)$ such that $E_i \cap E_j = \emptyset$ for $i \neq j$ and $\bigcup_{i=1}^{H} E_i = E$. For each $e \in E$ with weight $w_e \in w_{min} \cdot ((1-\epsilon)^{-j}, (1-\epsilon)^{-(j+1)}]$ assign e to set E^j, and for each edge e independntly, keep moving e to higher levels E^i, $i \geq j$, one level at a time while a coin with heads probability $(1-\epsilon)$ comes up heads. Thus, we have $\mathbf{Pr}[e \in E_i \text{ for some } i \geq j] = \min\{1, \frac{w_e}{w_{min}}(1-\epsilon)^j\}$. For $e \in E$ let the *level* of e, denoted by $l(e)$, be the unique index j such that $e \in E_j$, and let $l^*(e)$ denote the smallest j such that $w_{min}(1-\epsilon)^j \leq \min\{1, w_e q(e)\}$.

Let $q(e) = \hat{q}_{\kappa,\epsilon}(e), e \in E$ be the vector of sampling parameters. We first define

Algorithm 7: SAMPLE-SPANNER(G, q)

1: **for** $j = 1, \ldots, H$ **do**
2: Construct a spanner S_j on (V, E_j) of stretch at most κ.
3: For each $e \in S_j$, assign weight 0 to e if $l(e) < l^*(e)$, otherwise assign weight $1/q(e)$.
4: **end for**
5: Return the weighted collection $S_1 \cup \ldots \cup S_H$.

The sampling process takes form

Algorithm 8: SPANNER-SPARSIFY(G, q, ϵ, Z)

1: $q \leftarrow$ ESTIMATE$(G, \log n, \epsilon)$.
2: $Z \leftarrow \Theta(\log^3 n/((1-\epsilon)\epsilon^3))$
3: **for** $t = 1, \ldots, Z$ **do**
4: $\quad X_t \leftarrow$ SAMPLE-SPANNER(G, q)
5: **end for**
6: **return** $\frac{1}{Z}(X_1 + \ldots + X_H)$

We now prove the main property of our sampling process.

Lemma 120 *Let $q = \hat{q}_{\kappa,\epsilon}$. Then for $Z = \Omega(\log n)$ whp, i.e. except for a negligible part of the sample space, the sampling process in Algorithm 8 is stochastically dominated by the process of independently sampling an edge e with probability $\min\{1, w_e q(e)\}$ Z times (process B), and dominates the process of independently sampling each edge with probability $(1-\epsilon)\cdot\min\{1, w_e q(e)\}$ Z times(process A).*

Proof: Denote the spanner constructed by t-th invocation of SAMPLE-SPANNER at level $j \in [1 : H]$ by $S_{t,j}$. We denote the system of sets $E_1, \ldots, E_H$ in the t-th invocation of SAMPLE-SPANNER by E_j^t. For an edge $e \in E$ we write $l_t(e)$ to denote the level of e in E_t^j. We refer to an edge $e = (u, v) \in E$ as *free* at level j in invocation t if $d_{E_j^t}(u, v) > \kappa$. By Lemma 118, (3) we have that if $l_t(e) \geq l^*(e)$, then

$$\mathbf{Pr}[e \text{ is free in } S_{t,l_t(e)}] \geq \mathbf{Pr}[d_{G_{\hat{q}_{\kappa,\epsilon}(e)}}(u, v) > \kappa] \geq 1 - \epsilon,$$

where the probability is over the coin tosses determining the distribution of $e \in E$ among E_j^t.

Since the invocations of SAMPLE-SPANNER use independent randomness, we have by an application of Chernoff bounds, using the assumption that $Z = \Omega(\log n)$, that with probability at least, say, $1 - n^{-5}$ for each $e \in E$ such that $l_t(e) \geq l^*(e)$, one has that e is free in at least a $1 - \epsilon$ fraction of the spanners $S_{t,l_t(e)}, t = 1, \ldots, Z$.

Consider any set $W \subseteq E$ of edges. Each edge may be sampled by our process between 0 and Z times. We will show that for any $\{z_e \in [0 : Z] | e \in W\}$

$$\prod_{e \in W} ((1-\epsilon)q_\kappa)^{z_e} \leq \mathbf{Pr}[e \text{ is chosen } z_e \text{ times } \forall e \in W] \leq \prod_{e \in W} q_\kappa^{z_e}, \tag{5.12}$$

where we say that edge e is *chosen* if it is included in a spanner *and given positive weight.*

Indeed, consider the following sampling process. Take Z copies of the edge set E and for each edge first toss a coin to determine which of the Z spanner constructions it belongs to, and then toss coins to determine the level E_j that the edge belongs to. Note that since, by the argument above, whp each edge is free in at least a $(1-\epsilon)$ fraction of the spanners, with probability at least $(1-\epsilon)$ the edge belongs to a spanner construction in which it is free. In that case the edge necessarily belongs to the spanner S_j (since its endpoints are at distance greater than κ in $E_j \setminus \{e\}$), and hence is chosen

with probability at least $(1-\epsilon)q_\kappa$. Since the coin tosses for different edges are independent, we get that

$$\mathbf{Pr}[e \text{ is chosen } z_e \text{ times } \forall e \in W] \geq \prod_{e \in W} ((1-\epsilon)q_\kappa)^{z_e}.$$

On the other hand, the weight of an edge is 0 when the edge is at level larger than $l^*(e)$, and hence, since these coin tosses are independent for different edges, for any $W \subseteq E$,

$$\mathbf{Pr}[e \text{ is chosen } z_e \text{ times } \forall e \in W] \leq \prod_{e \in W} q_\kappa^{z_e}.$$

■

We have shown that our sampling process is sandwiched between two independent sampling processes which sample edges with probabilities $(1-\epsilon)Z \cdot \min\{1, w_e q_\kappa\}$ (process A) and $Z \cdot \min\{1, w_e q_\kappa\}$ respectively (process B).

We now obtain

Proof of Theorem 103: Set $Z = O(\log^3 n/((1-\epsilon)\epsilon^3))$. By Lemma 118 we have that

$$\min\{1, Z \cdot w_e \hat{q}_{\kappa,\epsilon}(e)\} \geq \Theta(\log n/\epsilon^2) \cdot \min\{1, w_e R_e\}$$

for all $e \in E$. Let G'_A denote the subgraph sampled by process A and let G'_B denote the subgraph sampled by process B, where process A gives weight $\frac{1}{(1-\epsilon)q(e)}$ to a chosen edge and B gives weight $\frac{1}{q(e)}$. Let G' denote the output of Algorithm 8. Then we have by Corollary 106

$$G'_A \in (1 \pm \epsilon)G, G'_B \in (1 \pm \epsilon)G$$

and by Lemma 120 we have

$$(1-\epsilon) \cdot G'_A \prec G' \prec G'_B,$$

and hence $G' \in (1 \pm O(\epsilon))G$. ■

Chapter 6

Sparsification in the edge deletion model

In this chapter we study the problem of graph sparsification on dynamic graph streams. Graph sparsification was introduced by Benczúr and Karger [16], who gave a near linear time procedure that takes as input an undirected graph G on n vertices and constructs a weighted subgraph H of G with $O(n \log n/\epsilon^2)$ edges such that the value of every cut in H is within a $1 \pm \epsilon$ factor of the value of the corresponding cut in G. A graph H that satisfies this property is called an ϵ-cut sparsifier of G. This algorithm has subsequently been used to speed up algorithms for a host of applications involving cuts and flows such as finding approximately minimum or sparsest cuts in graphs ([16, 58]), as well as other applications (e.g. [53]). Spielman and Teng introduced a stronger class of sparsifiers called spectral sparsifiers that preserve the Laplacian quadratic form [85]. Subsequent work has developed a number of efficient algorithms for constructing cut and spectral sparsifiers [16, 83, 13, 60, 31, 49] (one such algorithm was presented in Chapter 5).

The algorithms developed in [16, 83, 31, 49] take near-linear time in the size of the graph and produce very high quality sparsifiers but require random access to the edges of the input graph G, which is often prohibitively expensive in applications involving massive data sets. The streaming model of computation, which restricts algorithms to use a small number of passes over the input and space polylogarithmic in the size of the input, has been studied extensively in various application domains—see [75] for an overview—but has proven too restrictive for even the simplest graph algorithms. Even testing $s - t$ connectivity requires $\Omega(n)$ space [44]. The less restrictive semi-streaming model, in which the algorithm is restricted to use $\tilde{O}(n)$ space, is more suited for graph algorithms [27, 69].

The problem of constructing graph sparsifiers in the semi-streaming model was first considered by Ahn and Guha [2], who gave a one-pass algorithm for finding Benczúr-Karger type sparsifiers with a slightly larger number of edges than the original Benczúr-Karger algorithm, $O(n \log n \log \frac{m}{n}/\epsilon^2)$ as opposed to $O(n \log n/\epsilon^2)$ using $\tilde{O}(n)$ space. Subsequently, [57] obtained an algorithm for constructing stronger *spectral* sparsifiers of size $O(n \log n/\epsilon^2)$ in a single pass in the streaming model. All of these algorithms work only in the *incremental model*, where edges can be added to the graph but not removed.

In a recent paper [5] Ahn, Guha and McGregor introduced a beautiful graph sketching approach to streaming computations in dynamic streams, i.e., allowing both edge additions and deletions. They showed that connectivity can be determined in $\tilde{O}(n)$ space in this setting, and gave a *multi-pass* algorithm for obtaining cut sparsifiers in small space. Their techniques center around the use of linear sketches, which have been heavily studied in the field of compressed sensing/sparse recovery originating in [19, 25]. See [34] for a survey. Our focus in this chapter is on providing a *single-pass* implementation of cut sparsification on dynamic streams in the semi-streaming model.[1]

Our results: Our main result is an algorithm for constructing cut sparsifiers in a single pass in dynamic streams with edge deletions. We prove

Theorem 121 *There exists a single-pass streaming algorithm for constructing an ϵ-cut sparsifier of an unweighted, undirected graph $G = (V, E)$ with n vertices and m edges in the dynamic model*

[1] A similar result was independently and concurrently obtained by [6].

using $\tilde{O}(n/\epsilon^2)$ space. The size of the sparsifier is $O(n \log^3 n/\epsilon^2)$, and the runtime of the algorithm is $\tilde{O}(1/\epsilon^2)$ per update. At each point in the stream we can recover the edges of the sparsifier in time $\tilde{O}(n/\epsilon^2)$.

Our sparsification algorithm works by sampling edges at a rate inversely proportional to their edge connectivity, which was shown to work in [31]. In order to do this we maintain two sets of data structures. The first estimates connectivities, and the second does the actual sampling. We estimate connectivities by sampling edges of the input graph at a geometric sequence of sampling rates and recovering connected components of these samples using a result of [5]. The second set of data structures stores a linear sketch of the actual samples we use in our sparsifier, sampling edges incident on each vertex at a geometric sequence of rates. Using sparse recovery and the linearity of our sketch we are able to reconstruct the necessary samples when needed.

The main challenges that we need to overcome to make this work is (1) being able to estimate connectivities and produce edges of a sparsifier using random variables with limited indepence and (2) recovering the edges of the sparsifier from stored sketches. In order to put these challenges in perspective and outline our approach to overcoming them, suppose that G is k-connected and does not contain any $2k$-strongly connected components. The latter assumption implies that the *average* degree of G is $O(k)$. Edges of G can be sampled at rate $\tilde{O}(1/k)$ in this case, resulting in $\tilde{O}(1)$ edges incident on each vertex *on average*. If all vertex degrees are in fact close to average, one can show as in Section 6.3 that using sampling with $\tilde{\Theta}(1)$ independence per vertex is sufficient, and then the sampled edges can be recovered from linear sketches of edges at each vertex. However, complications arise when degrees are non-uniform: if a vertex in G has degree significantly above average, we cannot in general reconstruct its sampled edges from the sketch stored of the vertex, and it is no longer clear if this construction would actually yield a sparsifier, since the random variables used for sampling these edges do not have sufficient independence. Nevertheless, we give a recursive peeling-type decomposition of the graph that repeatedly removes low-degree nodes, reducing the degree of those that remain and allowing us to recover edges of the sparsifier from the sketches.

Organization: We start by giving preliminaries on graph sparsification in Section 6.1. We then describe the algorithm in Section 6.2 and Section 6.3. Maintaining our samples in the dynamic model requires knowing, for each edge in the graph, whether or not it was included in each sample. This can be easily achieved if we assume that our algorithm has access to $\tilde{\Theta}(n^2)$ independent random bits, which, however, is not feasible in $\tilde{O}(n)$ space. For simplicity of presentation, we first describe our algorithm assuming that it has access to $\tilde{\Theta}(n^2)$ random bits in Section 6.2. We show how to obtain sufficiently good estimates of edge connectivities in a single pass, as well as recover the edges of a sparsifier using sparse recovery techniques. In Section 6.3 we show how to remove the assumption that the algorithm has access to $\tilde{\Theta}(n^2)$ independent random bits using random variables with limited independence, obtaining a $\tilde{O}(n/\epsilon^2)$ space single-pass algorithm for sparsification in the dynamic model.

6.1 Sparsification preliminaries

We will denote by $G(V, E)$ the undirected input graph with vertex set V and edge set E with $|V| = n$ and $|E| = m$. For our purposes G will be unweighted, but the results in this section apply to weighted graphs. For any $\epsilon > 0$, we say that a weighted graph $G'(V, E')$ is an *ϵ-sparsification* of G if the weight of every cut in G' is within $(1 \pm \epsilon)$ of the corresponding cut in G.

Sparsification algorithms work by sampling edges with probabilities inversely proportional to some measure of connectivity. The simplest of these is edge-connectivity:

Definition 122 *A graph G is* k-connected *if the value of each cut in G is at least k, and an edge e has* edge-connectivity c_e *if c_e is the value of the minimum cut separating its endpoints.*

Graphs with high k-connectivity are particularly simple to sample:

Theorem 123 ([52]) *Let $G = (V, E)$ be a k-connected graph on n nodes, and let G' be obtained from G by sampling edges independently with probability $p = \Theta(\log n/(\epsilon^2 k))$, and giving sampled edges weight $1/p$. Then G' is an ϵ-sparsifier of G with high probability.*

The Benczúr-Karger algorithm samples according to a more strict notion of connectivity, referred to as *strong connectivity*, defined as follows:

Definition 124 ([16]) *A* k-strong component *is a maximal k-connected vertex-induced subgraph. The* strong connectivity *of an edge e, denoted by s_e, is the largest k such that a k-strong component contains e, and we say e is* k-strong *if its strong connectivity is k or more, and* k-weak *otherwise.*

The following two lemmas will be useful in our analysis:

Lemma 125 ([16]) *The number of k-weak edges in a graph on n vertices is bounded by $k(n-1)$.*

Lemma 126 ([16]) *Let $G = (V, E)$ denote an undirected graph on n nodes. For an edge $e \in E$ let s_e denote the strong connectivity of e. Then $\sum_{e \in E} 1/s_e \leq n - 1$.*

We also rely on Benczúr and Karger's main result, which is as follows:

Theorem 127 ([16]) *Let G' be obtained by sampling edges of G with probability $p_e = \min\{\rho/(\epsilon^2 s_e), 1\}$, where $\rho = 16(d+2)\ln n$, and giving each sampled edge weight $1/p_e$. Then G' is an ϵ-sparsification of G with probability at least $1 - n^{-d}$. Moreover, the expected number of edges in G' is $O(n \log n)$.*

It follows easily from the proof of Theorem (127) in [16] that if we over-sample by using an *underestimate* of edge strengths, the resulting graph is still an ϵ-sparsification.

Corollary 128 *Let G' be obtained by sampling each edge of G with probability $\tilde{p}_e \geq p_e$ and and give every sampled edge e weight $1/\tilde{p}_e$. Then G' is an ϵ-sparsification of G with probability at least $1 - n^{-d}$.*

Recently Fung *et al.*[31] proved that a more aggressive sampling method, namely sampling using edge connectivities as opposed to strong connectivities, also produces cut sparsifiers, and we will also require this result.

Theorem 129 ([31]) *Let G' be obtained from a weighted graph G by independently sampling edge e with probability $p_e = \rho/c_e$, where $\rho = \Theta(\log^2 n/\epsilon^2)$. Then, G' contains $O(n \log^2 n/\epsilon^2)$ edges in expectation and is an ϵ-sparsification whp.*

6.2 Sparsification with free randomness

In this section we present a dynamic sparsifier under the assumption that the algorithm has access to $\tilde{\Theta}(n^2)$ random words. We will remove this assumption in Section 6.3.

Our input graph G is undirected and unweighted. As in [5], we will use the following representation of G:

Definition 130 *Given an unweighted graph $G = (V, E)$, let A_G be the $n \times \binom{n}{2}$ matrix with entry $(u, (v, w)) \in [n] \times \binom{[n]}{2}$ and $v < w$ given by*

$$a_{u,(v,w)} = \begin{cases} 1 & \text{if } u = v \text{ and } (v, w) \in E \\ -1 & \text{if } u = w \text{ and } (v, w) \in E \\ 0 & \text{otherwise} \end{cases}$$

Updates to the graph G in the form of the addition or deletion of an edge arrive one at a time in a streaming fashion. An update cannot delete an edge that does not exist or add one that already does, but other than these restrictions the order is adversarial, and the stream can be arbitrarily long. We need to maintain a data structure using only $\tilde{O}(n)$ space that allows us to efficiently construct an ϵ-sparsifier of the current graph G after any sequence of updates. We will accomplish this using a collection of linear sketches of the rows of A_G.

Our algorithm has two components: the first will maintain an estimate of the connectivity of each edge and therefore its sampling rate (as discussed in Section 6.1), and the second will store the actual samples. The former uses the tools developed by Ahn *et al.*[5], and the latter is based on the technique of sparse recovery developed in the sketching and compressed sensing literature [34].

Before delving into the details, we elaborate on our use of randomness. In this section we assume that for each pair $(u, v) \in [n]^2$ the algorithm has access to a uniformly random number $h_{(u,v)} \in [0, 1]$. In fact, we will need $O(\log n)$ independent copies of these random numbers, which we will denote by $h^b_{(u,v)}$, $b = 1, \ldots, O(\log n)$. These random variables will be used to estimate sampling rates for edges of G. We will also assume access to independent random numbers $g^b_{(u,v)} \in [0, 1]$, $b = 1, \ldots, O(\log n)$, which we will use to determine a partition of the vertex set needed for sampling. Finally, we also assume access to independent random numbers $g^*_{(u,v)} \in [0, 1]$ which will be used to sample edges of the sparsifier. Once g, g^* and h are sampled, they are fixed for the duration of the algorithm. This is important for handling deletions, as it ensures that edges can be removed from exactly those sketches to which they have been added.

It will be convenient to think of all these numbers as independent in this section, even though this is not feasible in subquadratic space in the semi-streaming model. In Section 6.3 we will show that using numbers that are only $\tilde{O}(1)$-wise independent for fixed u and independent for different v is sufficient, leading to a space-efficient solution. Some of our subroutines will also require their own internal entropy, but the total used will be only $\tilde{O}(n)$ words.

6.2.1 Estimating edge connectivity

The building block of our connectivity estimates is the following result from [5] that finds connected components by sketching the rows of A_G:

Theorem 131 ([5]) *There is a single-pass, linear sketch-based algorithm supporting edge additions and deletions that uses $O(n \log^3 n)$ space and returns a spanning forest of the graph with high probability.*

Our sketch is simple. We consider random samples of the input graph at geometric sampling rates and find connected components in each sample. Specifically, for each $a = 1, \ldots, O(\log n)$ denote by $G_a^b = (V, E_a^b)$ a subsample of the edges of G obtained by setting

$$E_a^b = \left\{ (u,v) \in E : \min\{h_{(u,v)}^b, h_{(v,u)}^b\} < 2^{-a} \right\}. \tag{6.1}$$

We maintain connectivity data structures C_a^b for each of the subgraphs G_a^b for $a = 1, \ldots, O(\log n)$, $b = 1, \ldots, O(\log n)$ using Theorem 131. It is important to note that use of the functions $h_{(u,v)}$ for sampling rather than fresh randomness allows us to handle deletions properly by deleting a removed edge only from the sketches that we added it to by simply sampling and using the consistent samples as input to the sketch in Theorem 131.

Remark 132 *Note that an edge (u,v) is included in E_a^b if the minimum of $h_{(u,v)}^b$ and $h_{(v,u)}^b$ is smaller than 2^{-a}. This will be important for the proof of correctness for hash functions with limited independence in Section 6.3.*

The connectivity structure of the subgraphs G_a^b allows us to associate a sampling rate with each edge. For an edge $(u,v) \in E$ we use the smallest sampling rate 2^{-a} at which u and v are still in the same component as an estimate of the sampling rate for (u,v). Independent repetition $O(\log n)$ times reduces the variance sufficiently to get precise estimates.

Define $\mathcal{V}_a$ as the partition of vertices in V induced by the intersection of all partitions $C_a^b, b = 1, \ldots, O(\log n)$. That is, vertices u and v are in the same connected component in $\mathcal{V}_a$ if and only if they are connected in C_a^b for all $b = 1, \ldots, O(\log n)$. For an edge (u,v) let $L(u,v)$—the *level of* (u,v)—denote the largest a such that u and v are in the same component in $\mathcal{V}_a$, and for a vertex v let the *level* $L(v)$ denote the largest a such that v is not a singleton in $\mathcal{V}_a$.

The level $L(e)$ of an edge serves as a proxy for its connectivity:

Lemma 133 *For all edges $e \in E$ one has*

$$\Theta(s_e / \log n) \leq 2^{L(e)} \leq 2c_e$$

with high probability, where s_e denotes strong connectivity and c_e denotes edge connectivity.

Proof: The first inequality follows from the fact that $\Theta(\log n) \cdot 2^L$-strongly connected components will stay connected with high probability by Theorem 123 when the functions $h_{(u,v)}^b$ used for sampling

are truly random. Lemma 146 from Section 6.3 gives the result for hash functions with limited independence.

The second inequality follows by noting that if there is a cut separating u and v of size at most $2^L/2$, then it will be empty with probability at least $1/2$ when we sample at rate 2^{-L} by Markov's inequality, so u and v will get disconnected in one of the $O(\log n)$ independent repetitions with high probability. ■

This implies the levels can be used as sampling rates:

Lemma 134 *Sampling edges independently at rate $p_e = O\left(\log^2 n/(\epsilon^2 2^{L(e)})\right)$ and weighting sampled edges with $1/p_e$ produces a sparsifier with $O(n\log^3 n/\epsilon^2)$ edges high probability.*

Proof: By Theorem 129 sampling at rate $O(\log^2 n/(\epsilon^2 c_e))$ works. By Lemma 133 $1/2^{L(e)} \geq 1/(2c_e)$, and oversampling only improves concentration. This proves the statement assuming that the sampling of edges is independent.

The expected size of the sample is bounded by $\sum_{e\in E} p_e = O(\log^3 n/\epsilon^2)\cdot\sum_{e\in E} 1/s_e = O(n\log^3 n/\epsilon^2)$, where we used Lemma 133 to bound $p_e = O(\log n)/s_e$ and the fact that $\sum_{e\in E} 1/s_e \leq n-1$ by Lemma 126. ■

6.2.2 Maintaining edge samples

We now show how to maintain small space sketches that will allow us to reconstruct the edges of the sparsifier. Our basic tool is the technique of sparse recovery from the field of compressed sensing [34]. A vector A of dimension N is k-sparse if it has at most k non-zero entries, and a k-sparse, or approximately k-sparse, signal can be recovered with high probability from $O(k\log(N/k))$ non-adaptive linear measurements. Here we use the following result of Cormode and Muthukrishnan [24] that allows recovery in $\tilde{O}(k)$ time at the cost of slightly sub-optimal sketch size and error guarantees:

Theorem 135 ([24]) *We can construct a randomized $0/1$ matrix T of dimension $O(ck\log^3 n/\epsilon^2)\times N$ such that for any k-sparse signal A of dimension N, given the transformation TA we can reconstruct A exactly with probability at least $1-n^{-c}$ in time $O(c^2 k\log^3 n/\epsilon^2)$. The matrix T is constructed using $O(1)$-wise independent hash functions, and individual entries can be queried efficiently.*

We will also need the following result by Indyk [46] on sketching ℓ_1 norms:

Theorem 136 ([46]) *There is a linear sketch-based algorithm using $O(c\log^2 N/\epsilon^2)$ space and $O(c\log^2 N/\epsilon^2)$ random bits that can estimate the ℓ_1 norm of a vector of dimension N to within a factor of $(1\pm\epsilon)$ with probability $1-N^{-c}$. The sketch can be updated in $O(\log N)$ time.*

For a row v of the matrix A_G let $S_a^r(v)$ for $r = 1,\ldots,O(\log n)$ denote linear sketches guaranteed by Theorem 135 for $k = O(\log^3 n/\epsilon^2)$ where a corresponds to the geometric sequence of sampling rates, and $r = 1,\ldots,O(\log n)$ are independent copies that are useful for recovery. More precisely, $S_a^r(v)$ is a sketch of row v in the matrix $A_{G'_{a,r}}$ where $G'_{a,r}$ has edges

$$E'_{a,r} = \left\{(u,v)\in E : \min\{g^r_{(u,v)}, g^r_{(v,u)}\} < 2^{-a}\right\}. \tag{6.2}$$

Some of these sketches may accumulate more than k edges and consequently cannot be decoded on their own, but we will prove this is not an issue. We will also need sketches $d_a^r(v), r = 1, \ldots, O(\log n)$ for the ℓ_1-norm of the v-th row of the matrix $A_{G'_{a,r}}$. Note row v of $A_{G'_{a,r}}$ contains a ± 1 entry for each edge incident on v in $G'_{a,r}$, so the ℓ_1 norm corresponds exactly to the degree of v in the sampled graph $G'_{a,r}$.

Remark 137 *Note that we are using $O(\log n)$ independent samples $G'_{a,r}$ for each sampling rate a. This will be important in the proof of Lemma 141 below.*

Finally, we will also need another set of independent samples of G that will be used to obtain the edges of the sparsifier. Let G_a^* be the graph with edges

$$E_a^* = \left\{(u,v) \in E : \min\{g^*_{(u,v)}, g^*_{(v,u)}\} < 2^{-a}\right\}. \tag{6.3}$$

For each node v and sampling rate a we maintain sketches $S_a^*(v)$ of row v in the matrix $A_{G_a^*}$ using Theorem 135 for $k = O(\log^3 n/\epsilon^2)$. Here we do not need independent repetitions for each sampling rate a.

By the choice of the matrix A_G and the linearity of the sketches, if $\mathcal{S} \subseteq V$ is a cut then $\sum_{v\in\mathcal{S}} d_a^r(v)$ is a sketch for the size of the cut and $\sum_{v\in\mathcal{S}} S_a^r(v)$ is sketch of a sample of its edges. If v is a supernode obtained by contracting a set of vertices $\mathcal{S}$, we write $d_a^r(v)$ to denote $\sum_{u\in\mathcal{S}} d_a^r(u)$ and similarly for $S_a^r(v)$ and $S_a^*(v)$. For any fixed cut and fixed G'_a the estimate given by $d_a^r(v)$ is close to expectation with high probability.

Before specifying the algorithm formally, we give the intuition behind it. Recall that for every vertex u at level $L(u) = a$, we need to sample edges going from u to vertices in u's component in $\mathcal{V}_a$ with probability $\gamma \log^2 n/(\epsilon^2 2^a)$ for an appropriate constant γ. In order to do that, we will sample *all edges incident on u* with probability $\gamma \log^2 n/(\epsilon^2 2^a)$ and then throw away the ones that do not go to u's component. In order to obtain such a sample, we will use the sketches $S_{a'}^r(u)$ that were made with sampling at rate $\gamma \log^2 n/(\epsilon^2 2^a)$.

The main observation here is that if we contract connected components in $\mathcal{V}_{a+1}$ into supernodes, the resulting graph will have only $(\gamma \log n \cdot 2^{a+1})$-weak edges for a constant γ, so the average degree will be no larger than $\gamma \log n \cdot 2^{a+1}$. By repeatedly removing vertices with degree at most twice the average, nodes of this subgraph can be partitioned into sets $W_1, \ldots, W_t$ such that for each $i = 1, \ldots, t$ and $u \in W_i$ the degree of u in $W_i \cup \cdots \cup W_t$ is at most $4\gamma \log n 2^a$ and $t = O(\log n)$. Formally,

Lemma 138 *Suppose all edges in G are k-weak. Then V can be partitioned into $t = \log n$ sets $W_1, \ldots, W_t$ such that for all $v \in W_i$ the degree of v when restricted to $W_i \cup \cdots \cup W_t$ is at most $2k$.*

Proof: By Lemma 125 if G has n' nodes then it has at most $k(n'-1)$ edges. Let W_1 be the set of all nodes with degree at most $2k$. By Markov's inequality W_1 includes at least half the nodes. After removing W_1 and all incident edges we can repeat this process to find W_2, etc. At each iteration we remove at least half the nodes, so it terminates in $\log n$ iterations. ■

This partition cannot actually be computed because we cannot properly update the degree sketches after removing W_1, but its existence allows us to prove that the same procedure works when using the lower degree sample $G'_{a,r}$.

Let $\Delta = \log(\gamma \log^2 n/\epsilon^2)$. We need to use the samples $S^r_{a-\Delta}(u)$ for edges at level a. We first bound the degrees in $G'_{a-\Delta,r}$:

Lemma 139 *Let $g^r_{(u,v)}$ be $\Theta(\log^3 n/\epsilon^2)$-wise independent for fixed u, and independent for different u. Then the degree of all $u \in W_i$ in $G'_{a-\Delta,r}$ restricted to nodes $W_i \cup \ldots \cup W_t$ is at most $O(\log^3 n/\epsilon^2)$ with high probability.*

Proof: Consider a vertex $u \in W_i$ and let $N(u)$ denote the neighbors of u in $W_i \cup \ldots \cup W_t$ in the full graph G. The size of its sampled neighborhood is bounded by

$$\sum_{v \in N(u)} \mathbf{1}_{g^r_{(u,v)} < \gamma \log^2 n/(\epsilon^2 2^a)} + \sum_{v \in N(u)} \mathbf{1}_{g^r_{(v,u)} < \gamma \log^2 n/(\epsilon^2 2^a)}$$

The number of terms is $O(\log n 2^a)$. The second sum consists of independent random variables, so standard Chernoff bounds apply. Concentration bounds from Theorem 145 apply to the first sum since they are sufficiently independent for expectation $O(\log^3 n/\epsilon^2)$. ■

Observe that if a node u satisfies the bound in Lemma 139, then the sketch $S^r_{a-\Delta}(u)$ can be decoded using Theorem 135. Let U_1 be the set of decodable nodes. We would like to argue that we can simply output a decoded edge (u, v) as part of the sparsifier if and only if $g^r_{(u,v)} < \gamma \log^2 n/(\epsilon^2 2^a)$ and v belongs to the same connected component as u in $\mathcal{V}_a$. Then, using the linearity of the sketches, we could subtract decoded edges of the form $(u, v), u \in U_1, v \in V \setminus U_1$ from the sketches $S(v)$ and $d(v)$, effectively removing U_1 from the graph, and then move on to U_2.

However, for technical reasons to avoid dependencies and ensure the algorithm works in small space, we cannot reuse the variables $g^r_{(u,v)}$. We need to calculate U_2 using the independent sketches S^{r+1} and d^{r+1} as opposed to S^r and d^r to avoid dependencies and also use the variables $g^*_{(u,v)}$ for the actual samples. The size of r will remain bounded since we will prove the process terminates in $O(\log n)$ steps, but switching to S^{r+1} introduces additional complications because to remove a vertex $u \in U_1$ from the graph, we must be able to recover the edges from $S^{r+1}_{a-\Delta}(u), \ldots, S^{O(\log n)}_{a-\Delta}(u)$. The following lemma accomplishes this:

Lemma 140 *Let $g^r_{(u,v)}$ be $\Theta(\log^3 n/\epsilon^2)$-wise independent for fixed u, and independent for different u, and suppose the degree of u in $G'_{a-\Delta,r}$ is at most $\alpha \log^3 n/\epsilon^2$ where α is the constant from Lemma 139. Then the degree of u in $G'_{a-\Delta,r'}$ is $O(\log^3 n/\epsilon^2)$ for all $r' \geq r$ with high probability.*

Proof: If the degree of u in $G'_{a-\Delta,r}$ is at most $\alpha \log^3 n/\epsilon^2$, we expect the degree in G to be at most $(\alpha/\gamma) \log n 2^a$, and concentration inequalities for sampling with limited independence (Theorem 145) show that with high probability its degree is at most, say, $2(\alpha/\gamma) \log n 2^a$. Applying Theorem 145 again shows u's degree in $G'_{a-\Delta,r'}$ is $O(\log^3 n/\epsilon^2)$ for any r', and taking a union bound over all r' finishes the proof. ■

We now state the algorithm formally. For each $a = 1, \ldots, O(\log n)$ we denote the graph obtained by contracting all connected components in $\mathcal{V}_{a+1}$ into supernodes by H_a.

Algorithm 9: PARTITION(a)

1: Let $H_a^1 \leftarrow H_a$.
2: **for** $r \leftarrow 1$ to $O(\log n)$ **do**
3: Estimate the degree of each $v \in H_a^r$ from sketches $d_{a-\Delta}^r(v)$
4: $U_a^r \leftarrow \{v \in H_a^r | d_{a-\Delta}^r(v) \leq 4\alpha \log^3 n/\epsilon^2\}$
5: **for** $u \in U_a^r$, $j = r+1, \ldots, O(\log n)$ **do**
6: Run sparse recovery on $S_{a-\Delta}^j(u)$
7: For all edges (u, v) recovered from $S_{a-\Delta}^j(u)$, subtract (u, v) from $S_{a-\Delta}^j(v)$ and $d_{a-\Delta}^j(v)$
8: **end for**
9: $H_a^{r+1} \leftarrow H_a^r \setminus U_a^r$.
10: **end for**
11: **return** $\{U_a^r\}_{r=1,\ldots,O(\log n)}$

Here γ is a constant such that sampling the edges of a k-connected graph at rate $\gamma \log n/k$ produces a connected subgraph with probability at least $1 - n^{-10}$, and α is a constant bounding the degree in Lemma 139.

We first prove

Lemma 141 *For all $a = 1, \ldots, O(\log n)$, Algorithm 9 recovers a partition of H_a such that for all $r = 1, \ldots, O(\log n)$ for each $u \in U_a^r$ the degree of u in $U_a^{r+1} \cup \ldots \cup U_a^{O(\log n)}$ in graph $G_{a-\Delta}^*$ is $O(\log^3 n/\epsilon^2)$ with high probability.*

Proof:

We first prove that the constructed sets cover all of H_a. Consider the set U_1. By Lemma 139, all $u \in W_1$ have degree $O(\log^3 n/\epsilon^2)$ in $G'_{a-\Delta,1}$ with high probability, so removing all nodes with degree at most $4\alpha \log^3 n/\epsilon^2$ for large enough α will include all $u \in W_1$. Lemma 140 implies that the sparse recovery in line 6 will succeed for all j with high probability, so we can completely remove U_a^r from the graph. Lemma 140 also bounds the degree of $u \in U_a^r$ in graph $G_{a-\Delta}^*$, by replacing $G'_{a-\Delta,r'}$ with $G_{a-\Delta}^*$ in the statement of the lemma.

We now note that the identity of the set U_a^r is independent of the randomness used for samples and sketches $S_{a-\Delta}^j$, $d_{a-\Delta}^j$, $j = r+1, \ldots, O(\log n)$. Thus, the same bounds on node degrees follow by a recursive application of the argument to $H_a \setminus U_a^1$. Furthermore, it follows by induction on r that after removing $U_a^1, \ldots, U_a^r$ we have removed all of $W_1, \ldots, W_r$ with high probability, so the algorithm terminates in $O(\log n)$ iterations. ■

Given the partition $U_a^1, \ldots, U_a^{O(\log n)}$, the algorithm for recovering the edges of the sparsifier is as follows:

Algorithm 10: RECOVER(a)

1: **for** $r = 1, \ldots, O(\log n)$, $u \in U_a^r$ **do**
2: Run sparse recovery on $S^*_{a-\Delta}(u)$
3: Output each recovered edge $(u, v), u \in U_r$ only if $g^*_{(u,v)} < \gamma \log^2 n/(\epsilon^2 2^a)$ and $L(u, v) = a$.
4: Subtract recovered edges from $S^*_{a-\Delta}(v)$ for all $v \in U_{a,r+1} \cup \ldots U_{a,O(\log n)}$.
5: **end for**

We can now prove

Theorem 142 *For each $v \in V(G)$ Algorithm 10 recovers a sample of edges incident on v, where edges are picked with probability $\gamma \log^2 n/(\epsilon^2 2^{L(v)})$.*

Proof:

Note that sparse recovery succeeds whp by the degree bound in Lemma 141. Finally, note that the structure of the partition $U_1 \cup \ldots \cup U_r$ maps to each edge a single random variable $g_{(u,v)}$, so the probability of an edge being sampled is correct. ■

We will need the following definition in Section 6.3:

Definition 143 *An edge $e = (u, v) \in E$ is controlled by a vertex $u \in V$ if e is sampled using $g^*_{(u,v)}$. We denote the set of edges controlled by u by E_u.*

We will also need

Lemma 144 *Let E^* be a set of edges. For each $u \in V$ one has $\mathbf{E}[|E^* \cap E_u|] = O(\log^4 n/\epsilon^2)$.*

Proof: Consider a vertex $u \in V$. By Lemma 141 u controls $O(\log^3 n 2^a/\epsilon^2)$ edges at level a. Hence, the expected number of edges sampled at each level a is $O(\log^3 n/\epsilon^2)$. Hence, the expected number of edges controlled by u across all levels is $O(\log^4 n/\epsilon^2)$. ■

6.2.3 Runtime

We now briefly summarize the time required to update the sketches and to construct a sparsifier. We do not optimize the log factors but only show updates require $\tilde{O}(1/\epsilon^2)$ time and building a sparsifier requires $\tilde{O}(n/\epsilon^2)$. Each addition of deletion of an edge requires updating C_a^b, S_a^r and d_a^r for $a, b, r \leq O(\log n)$. The sketches C_a^b are built using ℓ_0-samplers (see [5]) and can be updated in $\tilde{O}(1)$ time. For an edge (u, v) we update all $O(\log n)$ copies of $S(u)$, $S(v)$, $d(u)$ and $d(v)$. By Theorems 135 and 136, these can each be updated in $\tilde{O}(1/\epsilon^2)$ time. For $S(u)$ this is done by querying only the $\tilde{O}(1/\epsilon^2)$ entries of the matrix T we need.

Construction of the sparsifier is more expensive. If we query the sparsifier after each graph update it may need to be recomputed from scratch each time due to edge deletions, so we cannot amortize its cost across the updates. However, we will show it requires only $\tilde{O}(n/\epsilon^2)$ time. Building all $O(\log n)$ $\mathcal{V}_a$ requires $\tilde{O}(n)$ operations each if ℓ_0-sampling is done efficiently.

Running one iteration of Algorithm 9 requires $O(n)$ estimations of d, $O(n)$ decodings of S, $\tilde{O}(k)$ updates to S and d for each of the $O(n)$ sparse recoveries and $O(n)$ additional bookkeeping. Since S is $\tilde{O}(1/\epsilon^2)$-sparse by Theorem 135 decoding takes $\tilde{O}(1/\epsilon^2)$ time. Summing over $\tilde{O}(1)$ values of r and a, we use a total of $\tilde{O}(n/\epsilon^2)$ time. Algorithm 10 also does $O(n)$ sparse recoveries and $\tilde{O}(n/\epsilon^2)$ updates to S^* per iteration, which also totals to $\tilde{O}(n/\epsilon^2)$ summing over all r and a.

6.2.4 Weighted graphs

We note that even though we stated the algorithm for unweighted graphs, the following simple reduction yields a single pass dynamic sparsifier for weighted graphs, as long as when an edge is removed or updated, its weight is given together with the identity of its endpoints. Suppose that edge weights are integers between 1 and W (the general case can be reduced to this one with appropriate scaling and rounding). Consider graphs $G_0, \ldots, G_{\log_2 W}$, where an edge $e = (u, v)$ belongs to the edge set of G_b iff the binary expansion of w_e has 1 in position b, for $b = 0, \ldots, \log_2 W$. Note that in order to preserve cuts in G to a multiplicative factor of $1 \pm \epsilon$, it is sufficient to preserve cuts in each of $G_0, \ldots, G_b$ to the same factor. To do that, it is sufficient to maintain $\log_2 W$ copies of our algorithm operating on the graphs G_b (this is feasible due to the assumption that edges are either added or completely removed, i.e., the weight of the removed edge is given at the time of removal). The space used and the number of edges in the sparsifier will both increase by a factor of $\log_2 W$.

6.3 Sparsification with limited independence

In this section we remove the assumption that the algorithm has access to $\tilde{\Theta}(n^2)$ bits of randomness by using sampling with limited independence. We prove the following two statements. First, we show in Lemma 146 that sampling edges of a k-connected graph at rate $\gamma \log n/k$ yields a connected graph with high probability, *even with limited independence*. In particular, it is sufficient to ensure that random variables used for sampling edges incident to any given vertex are only $\tilde{O}(1/\epsilon^2)$-wise independent. This lemma is used in Section 6.2 to show that our estimation of sampling rates is accurate. We then show that our algorithm for constructing a sparsifier by sampling at rates proportional to edge connectivities yields a sparsifier with high probability even when the sampling is done using limited independence. We note that the second claim does not subsume the first due an extra $\log n$ factor that is needed for sampling with edge connectivities to go through.

We will use tail bounds for t-wise independent random variables proved in [79], Theorem 5:

Theorem 145 *Let $X_1, \ldots, X_n$ be random variables each of which is confined to $[0,1]$. Let $X = \sum_{i=1}^{n} X_i, \mu = \mathbf{E}[X]$. Let $p = \mu/n$, and suppose that $p \leq 1/2$. Then if X_i are $\lceil \epsilon\mu \rceil$-wise independent, then*

$$\mathbf{Pr}[|X - \mu| \geq \epsilon\mu] < e^{-\lfloor \epsilon^2 \mu/3 \rfloor},$$

if $\epsilon < 1$, and

$$\mathbf{Pr}[|X - \mu| \geq \epsilon\mu] < e^{-\epsilon \ln(1+\epsilon)\mu/2} < e^{-\epsilon\mu/3}$$

otherwise.

We now prove

Lemma 146 *Let $G = (V, E)$ be a k-connected graph on n nodes. For edges $e = (u, v) \in E$ let random numbers $h_{u,v} \in [0,1]$ be such that*

1. *$h_{u,v}$ is independent of $h_{u',v'}$ for all $u' \neq u$;*
2. *$h_{u,v}$ are $\lceil 4\gamma \log n \rceil$-wise independent for fixed u, where $\gamma \geq 20$.*

Also, let $X_{u,v}$ be 0/1 random variables such that $X_{u,v} = 1$ if $h_{u,v} \leq (\gamma \log n)/k$ and 0 otherwise. If G' is obtained by including each edge $(u, v) \in E$ such that $X_{u,v} = 1$ or $X_{v,u} = 1$, then G' is connected whp.

Proof: First, for each $e = (u, v) \in E$ let $\hat{X}_{u,v}$ denote 0/1 random variables such that $X_{u,v} = 1$ if $h_{u,v} < (\gamma \log n)/s_e$, where s_e is the strong connectivity of e. Define $\hat{G}'$ as the graph obtained by including each edge $(u, v) \in E$ such that $\hat{X}_{u,v} = 1$ or $\hat{X}_{v,u} = 1$. Note that $\hat{G}'$ is a subgraph of G', so it is sufficient to show that $\hat{G}'$ is connected whp.

Suppose that $F = (V_F, E_F)$ is a k-connected graph without $2k$-strongly connected components for some k. Recall from Lemma 138 that V_F can be partitioned into $\log |V_F|$ sets $W_1, \ldots, W_{\log |V_F|}$ such that the degree of any $u \in W_r$ in $W_r \cup \cdots \cup W_{\log |V_F|}$ is at most $4k$. For each node $u \in W_r$ let E_u denote the edges incident on u that go to nodes in $W_r \cup \cdots \cup W_{\log |V_F|}$ (if an edge $e = (u, v)$

goes between two nodes in W_r, include it either in E_u or E_v arbitrarily). We will say vertex $u \in W_r$ *controls* edges $e \in E_u$. Note that $|E_u| \leq 4k$ for all $u \in W_r$, and hence the expected number of edges sampled in E_u is at most $4\gamma \log n$. Note that this definition of control is slightly different from the one given in Definition 143. In particular, this is because Definition 143 pertains to the actual sampling procedure that our algorithm uses, while here we are concerned with the estimation step.

Let $j_{max} = \lfloor \log_2 n \rfloor$. We will show by induction on $j = j_{max}, \ldots, 0$ that all 2^j-connected components are connected with probability at least $1 - (j_{max} - j + 1)n^{-3}$.

Base: $j = j_{max}$ We have $\kappa = 2^{j_{max}}$. Apply the decomposition above to the κ-strongly connected components of G, which does not have any 2κ-connected components since $2\kappa > n$. Let $\hat{G}''$ denote the subgraph of G obtained by including for each $u \in V$ edges $e = (u, v) \in E_u$ when $X_{u,v} = 1$. Denote the set of sampled edges by E'. Recall that for all $u \in V(G)$ one has $\mathbf{E}[|E' \cap E_u|] \leq 4\gamma \log n$.

Fix a cut $(C, V \setminus C)$. For each vertex $u \in C$ let $X_u = \sum_{(u,v) \in E_u, v \notin C} X_{u,v}$.

By setting $\epsilon = 1$ in Theorem 145 we get

$$\mathbf{Pr}[X_u = 0] < e^{-\mathbf{E}[X_u]/3}.$$

Since $X_u, X_{u'}$ are independent for $u \neq u'$, the probability that the cut is empty is at most

$$\prod_{u \in C} e^{-\mathbf{E}[X_u]/3} = e^{-\gamma |C| \log n/(3k)}.$$

By Karger's cut counting lemma, the number of cuts of value at most αk is at most $n^{2\alpha}$. A union bound over all cuts, we get failure probability at most

$$\sum_{\alpha \geq 1} n^{2\alpha} e^{-\gamma \alpha \log n/3} \leq n^{-4}$$

since $\gamma \geq 20$. Taking a union bound over all κ-connected components yields failure probability at most n^{-3}.

Inductive step: $j + 1 \to j$ We have $\kappa = 2^j$. By the inductive hypothesis, all 2^{j+1}-connected components will be connected with probability at least $1 - (j_{max} - (j + 1) + 1)n^{-3}$. We condition on this event and contract the connected components into supernodes.

We now have a union of vertex-disjoint κ-strongly connected components that do not contain any 2κ-connected components. The same argument as in the base case shows that each such component will be connected with probability at least $1 - n^{-4}$. A union bound over at most n such components completes the inductive step.

■

In order to show that the results of [31] carry over to our setting, it is sufficient to show that the following version of Chernoff bounds holds under our limited independence assumptions (Theorem 2.2 in [31]):

Theorem 147 *Let $X_1, \ldots, X_n$ be n random variables such that X_i takes value $1/p_i$ with probability p_i and 0 otherwise.Then, for any p such that $p \leq p_i$, for each i, any $\epsilon \in (0,1)$ and any $N \geq n$ the following holds:*

$$\mathbf{Pr}\left[\left|\sum_{i=1}^{n} X_i - n\right| > \epsilon N\right] < 2e^{-0.38\epsilon^2 pN}.$$

Indeed, an inspection of the proofs of Lemma 4.1 and Lemma 5.5 in [31] shows that the authors (a) only rely on independence of their sampling process to obtain Theorem 147 and (b) only apply Theorem 147 to subsets of edges of G, where p_i are sampling probabilities.Thus, proving an equivalent of Theorem 147 allows us to extend their results to our limited independence sampling approach.

We now prove

Lemma 148 *Let $G = (V, E)$ denote an unweighted undirected graph. Let $\gamma > 0$ be a sufficiently large constant such that sampling at rate $\gamma \log^2 n/c_e$ independently produces a sparsifier with probability at least $1 - n^{-2}$, where c_e is the edge connectivity of e. Let $X_e, e \in E$ be random variables corresponding to including edges from a set E^* into the sample such that X_e takes value $1/p_e$ with probability p_e and 0 otherwise, where p_e is the sampling probability used by Algorithm 10. Assume that sampling is $c\log^4 n/\epsilon^2$-wise independent for a sufficiently large constant $c > 0$ that may depend on γ.*

There exists an event $\mathcal{E}$ with $\mathbf{Pr}[\mathcal{E}] > 1 - n^{-2}$ such that for any $E^* \subseteq E$, any $p \leq p_e, e \in E^*$, any $\epsilon \in (0,1)$ and any $N \geq |E^*|$

$$\mathbf{Pr}\left[\left|\sum_{e \in E^*} X_e - |E^*|\right| > \epsilon N | \mathcal{E}\right] < e^{-\epsilon^2 pN/6}.$$

Proof: For simplicity of exposition, we now assume that G is unweighted. Let $X_1, \ldots, X_n$ be random variables corresponding to picking edges of the graph. Recall that our sampling algorithm samples an edge (u, v) either depending on the value of $g_{(u,v)}$ or the value of $g_{(v,u)}$ (the choice depends on the partition of the node set $U_1 \cup \ldots \cup U_r$ constructed in Algorithm 10). Recall that by Definition 143 a node u *controls* edge (u, v) if Algorithm 10 samples (u, v) using the value of $g^*_{(u,v)}$. Note that each edge is controlled by exactly one node. For a node u, as before, let E_u denote the set of edges controlled by u. By Lemma 144, one has $\mathbf{E}[|E^* \cap E_u|] = O(\log^4 n/\epsilon^2)$. Let $\mathcal{E}$ denote the event that at most $2\gamma \log^4 n/\epsilon^2$ edges controlled by u are sampled, for all $u \in V$, where we are assuming that γ is sufficiently large. Since our random variables are $c\log^4 n/\epsilon^2$-wise independent for sufficiently large c, by Theorem 145 and a union bound over all u one has $\mathbf{Pr}[\mathcal{E}] \geq 1 - n^{-2}$.

For each $e \in E$ let X_e be a Bernoulli random variable that takes value p/p_e if edge e is sampled, and 0 otherwise, so that $X_e \in [0, 1]$.

Consider a set of edges E^*. Partition E^* as $E^* = \bigcup_{u \in V} E^*_u$, where $E^*_u = E^* \cap E_u$. Thus, random variables $\mathcal{X}_u := \sum_{e \in E^*_u} X_e$ are independent for different i. Let $\mathcal{X} = \sum_{u \in V} \mathcal{X}_u$, $\mu = \mathbf{E}[\mathcal{X}]$.

Then by Markov's inequality

$$\mathbf{Pr}[\mathcal{X} \geq (1+\delta)\mu|\mathcal{E}] \leq \frac{\mathbf{E}[e^{t\mathcal{X}}|\mathcal{E}]}{e^{t(1+\delta)\mu}}. \tag{6.4}$$

Recall that

$$\mathbf{E}[e^{t\mathcal{X}}|\mathcal{E}] = \sum_{j=0}^{\infty} \mathbf{E}[(t\mathcal{X})^j|\mathcal{E}]/j! = \sum_{j=0}^{\infty} t^j/j! \sum_{S\subseteq E^*, |S|\leq j} \sum_{\alpha_e \geq 0, \sum_{e\in S}\alpha_e = j} \mathbf{E}\left[\prod_{e\in S} X_e^{\alpha_e}|\mathcal{E}\right] \tag{6.5}$$

For any non-negative random variably $\mathcal{Y}$ one has

$$\mathbf{E}[\mathcal{Y}|\mathcal{E}] \leq \mathbf{E}[\mathcal{Y}]/\mathbf{Pr}[\mathcal{E}].$$

Conditional on $\mathcal{E}$, one has $\prod_{e\in S} X_e^{\alpha_e} = 0$ for all $S \subseteq E^*$ such that $|S \cap E_u| > 2\gamma \log^4 n/\epsilon^2$ for at least one $u \in V$. For other S, setting $\mathcal{Y} = \prod_{e\in S} X_e^{\alpha_e}$, one gets

$$\mathbf{E}\left[\prod_{e\in S} X_e^{\alpha_e}|\mathcal{E}\right] \leq \mathbf{E}\left[\prod_{e\in S} X_e^{\alpha_e}\right]/\mathbf{Pr}[\mathcal{E}]. \tag{6.6}$$

Combining (6.6) and (6.5) one gets

$$\mathbf{E}[e^{t\mathcal{X}}|\mathcal{E}] \leq \frac{1}{\mathbf{Pr}[\mathcal{E}]} \sum_{j=0}^{\infty} t^j/j! \sum_{S\subseteq E^*, |S|\leq j, |S\cap E_u|\leq 2\gamma\log^4 n/\epsilon^2} \sum_{\alpha_e \geq 0, \sum_{e\in S}\alpha_e = j} \mathbf{E}\left[\prod_{e\in S} X_e^{\alpha_e}\right] \tag{6.7}$$

On the other hand, for all $S \subseteq E^*$ such that $|S \cap E_u| \leq 2\gamma \log^4 n/\epsilon^2$ one has

$$\mathbf{E}\left[\prod_{e\in S} X_e^{\alpha_e}\right] = \prod_{e\in S} \mathbf{E}[X_e^{\alpha_e}]$$

by $\gamma \log^4 n/\epsilon^2$-wise independence. Thus, we get

$$\mathbf{Pr}[\mathcal{X} \geq (1+\delta)\mu|\mathcal{E}] \leq \frac{1}{\mathbf{Pr}[\mathcal{E}]} \frac{\prod_{e\in E^*} \mathbf{E}[e^{tX_e}]}{e^{t(1+\delta)\mu}}, \tag{6.8}$$

which is the same bound as in the full independence case, except for a factor of $1/\mathbf{Pr}[\mathcal{E}] = 1+O(1/n)$ in front. Now the same derivation as in the full independence case shows that the probability of overestimating is appropriately small.

We now bound the probability of underestimating. Consider a set of edges E^*. Partition E^* as $E^* = \bigcup_{i=1}^{s} E_i$, where $E_i \cap E_j = \emptyset, i \neq j$, so that

1. $\mathbf{E}[\sum_{e\in E_i} X_e] \leq c \log^4 n/\epsilon^2$ for a sufficiently large constant $c > 0$;
2. random variables $\sum_{e\in E_i} X_e$ are independent for different i;

3. $s \leq \frac{\epsilon^2}{6\log(4/\epsilon^2)}\mathbf{E}[\sum_{e\in E^*} X_e]$.

Note that this is feasible since our graphs are unweighted, so ϵ can be assumed to be larger than $1/n^2$. For each $i = 1, \ldots, s$ let $\mathcal{X}_i := \sum_{e\in E_i} X_e$. Note that $\mathcal{X}_i$ are independent, and X_e are $c\log^4 n/\epsilon^2$-wise independent. Now by Theorem 145 for all $i = 1, \ldots, s$ one has for all $\epsilon \in (0, 1)$

$$\mathbf{Pr}[\mathcal{X}_i < \mathbf{E}[\mathcal{X}_i] - \epsilon\mathbf{E}[\mathcal{X}_i]] < e^{-\epsilon^2\mathbf{E}[\mathcal{X}_i]/3} \tag{6.9}$$

Let $\mathcal{X} = \sum_{i=1}^{s} \mathcal{X}_i$. For constant $\epsilon > 0$ let

$$\mathcal{K}(\epsilon) = \left\{ \mathbf{z} = (z_1, z_2, \ldots, z_s) \in \left\{0, \frac{1}{4}\epsilon^2, \frac{1}{2}\epsilon^2, \frac{3}{4}\epsilon^2, \ldots, 1-\epsilon^2/4, 1\right\}^s : \sum_{i=1}^{s} z_i\mathbf{E}[\mathcal{X}_i] \geq \epsilon\mathbf{E}[\mathcal{X}] \right\}.$$

We now have

$$\begin{aligned} \mathbf{Pr}\left[\mathcal{X} < \mathbf{E}\left[\mathcal{X}\right] - \epsilon\mathbf{E}[\mathcal{E}]\right] &\leq \sum_{\mathbf{z}\in\mathcal{K}(\epsilon)} \prod_{i=1}^{s} \mathbf{Pr}[\mathcal{X}_i < \mathbf{E}[\mathcal{X}_i] - (z_i - \epsilon^2/4)^2\mathbf{E}[\mathcal{X}_i]] \\ &\leq \sum_{\mathbf{z}\in\mathcal{K}(\epsilon)} \prod_{i=1}^{s} \mathbf{Pr}[\mathcal{X}_i < \mathbf{E}[\mathcal{X}_i] - z_i^2\mathbf{E}[\mathcal{X}_i] + (\epsilon^2/2)\mathbf{E}[\mathcal{X}_i]] \end{aligned} \tag{6.10}$$

since every set of values for $\mathcal{X}_i - \mathbf{E}[\mathcal{X}_i]$ such that $\sum_i(\mathcal{X}_i - \mathbf{E}[\mathcal{X}_i]) < -\epsilon\mathbf{E}[\mathcal{X}]$ can be rounded to a point in $\mathcal{K}(\epsilon)$ with a loss of at most $(\epsilon^2/2)\mathbf{E}[\mathcal{X}_i]$ in each term. We now note that for any $\mathbf{z} \in [0, 1]^s$ such that $\sum_{i=1}^{s} z_i\mathbf{E}[\mathcal{X}_i] = \epsilon'\mathbf{E}[\mathcal{X}] \geq \epsilon\mathbf{E}[\mathcal{X}]$ one has $\sum_{i=1}^{s} z_i^2\mathbf{E}[\mathcal{X}_i] \geq (\epsilon')^2\mathbf{E}[\mathcal{X}] \geq \epsilon^2\mathbf{E}[\mathcal{X}]$.

Next, since $s \leq \frac{\epsilon^2}{6\log(4/\epsilon^2)}(\sum_{i=1}^{s} \mathbf{E}[\mathcal{X}_i])$, we have that

$$\mathbf{Pr}\left[\mathcal{X} < \mathbf{E}\left[\mathcal{X}\right] - \epsilon\mathbf{E}[\mathcal{X}]\right] \leq (4/\epsilon^2)^s e^{-\epsilon^2\mathbf{E}[\mathcal{X}]/3} \leq e^{-\epsilon^2\mathbf{E}[\mathcal{X}]/6}. \tag{6.11}$$

■

Theorem 149 *The set of edges returned by Algorithm 10 is a sparsifier whp.*

Proof: Lemma 148 can be used instead of Theorem 147 in [31]. ■

Chapter 7

Matchings in regular bipartite graphs in $\tilde{O}(n^{1.5})$ time

In this chapter we present an algorithm for finding perfect matchings in regular bipartite graphs whose runtime improves significantly on the runtime of the sampling based algorithm presented in Chapter 2. Our main techniques here are a more general two-stage non-uniform sampling scheme and a specialized analysis of the runtime of the Hopcroft-Karp algorithm on the subsampled graph. The runtime of the resulting algorithm matches (up to polylogarithmic terms) the lower bound for algorithms that use uniform sampling to access the graph that we proved in Chapter 2.

We present a significantly faster algorithm for finding perfect matchings in regular bipartite graphs.

Theorem 150 *There is an* $O\left(\min\{m, \frac{n^2 \ln^3 n}{d}\}\right)$ *expected time algorithm to find a perfect matching in a* d*-regular bipartite graph* G*.*

As a function of n alone, the running time stated above is $O((n \ln n)^{1.5})$. Since the $O(m)$ running time is guaranteed by the algorithm of Cole, Ost, and Schirra, we are only concerned with the case where d is $\Omega(\sqrt{n} \ln n)$. For this regime, our algorithm reduces the perfect matching problem on a regular bipartite graph G to the same problem on a (not necessarily regular) sparse bipartite graph H with $O(n \ln n)$ edges. This reduction takes time $O(\frac{n^2 \ln^3 n}{d})$. We then use the Hopcroft-Karp algorithm on H to recover a perfect matching. A black-box use of the analysis of the Hopcroft-Karp algorithm would suggest a running time of $O(\frac{n^2 \ln^3 n}{d} + n^{1.5} \ln n)$. However, we show that the final sampled graph has some special structure that guarantees that the Hopcroft-Karp algorithm would complete in time $O(\frac{n^2 \ln^2 n}{d})$ whp.

For every pair $A \subseteq P, B \subseteq Q$, we define a *witness set* $W(A, B)$ to be the set of all edges going from A to $Q \setminus B$. Of particular interest are what we call *Hall witness sets*, which correspond to $|A| > |B|$; the well-known Hall's theorem [18] says that a bipartite graph $H(P, Q, E_H)$ contains a perfect matching iff E_H includes an edge from each Hall witness set. Thus any approach that reduces the size of the input bipartite graph by sampling must ensure that some edge from every Hall witness set is included in the sampled graph; otherwise the sampled graph no longer contains a perfect matching. In Chapter 2 we showed that no *uniform sampling* scheme on a d-regular bipartite graph can reduce the number of edges to $o(\frac{n^2}{d \ln n})$ while preserving a perfect matching, and hence their $\tilde{O}(n^{1.75})$-time algorithm is the best possible running time achievable via uniform sampling followed by a black-box invocation of the Hopcroft-Karp analysis.

In order to get past this barrier, we use here a two-stage sampling process. The first stage is a uniform sampling (along the lines of Chapter 2) which generates a reduced-size graph $G' = (P, Q, E')$ that preserves not only a perfect matching but also a key relationship between the sizes of "relevant" witness sets and cuts in the graph G. The second stage is to run the non-uniform Benczúr-Karger sampling scheme [16] on G' to generate a graph G'' with $\tilde{O}(n)$ edges while preserving a perfect matching w.h.p. Since this step requires $\tilde{\Omega}(|E'|)$ time, we crucially rely on the fact that G' does not contain too many edges.

While our algorithm is easy to state and understand, the proof of correctness is quite involved. The Benczúr-Karger sampling was developed to generate, for any graph, a weighted subgraph with $\tilde{O}(n)$ edges that approximately preserves the size of all cuts in the original graph. The central

idea underlying our result is to show that there exists a collection of *core* witness sets that can be identified in an almost one-one manner with cuts in the graph such that the probability mass of edges in each witness set is comparable to the probability mass of the edges in the cut identified with it. Further, every witness set in the graph has a "representative" in this collection of core witness sets. Informally, this allows us to employ cut-preserving sampling schemes such as Benczúr-Karger as "witness-preserving" schemes. We note here that the natural mapping which assigns the witness set of a pair (A, B) to the cut edges associated with this pair can map arbitrarily many witness sets to the same cut and is not useful for our purposes. One of our contributions is an uncrossing theorem for witness sets, that we refer to as the *proportionate uncrossing theorem*. Informally speaking, it says that given any collection of witness sets $\mathcal{R}$ such that the probability mass of each witness set is comparable to that of its associated cut, there exists another collection $\mathcal{T}$ of witness sets such that (i) the natural mapping to cuts as defined above is *half-injective* for $\mathcal{T}$, that is, at most two witness sets in $\mathcal{T}$ map to any given cut, (ii) the probability mass of each witness set is comparable to the probability mass of its associated cut, and (iii) any subset of edges that hits every witness set in $\mathcal{T}$ also hits every witness set in $\mathcal{R}$. The collection $\mathcal{T}$ is referred to as a proportional uncrossing of $\mathcal{R}$. As shown in Figure 7.1(a), we can not achieve an injective mapping, and hence the half-injectivity is unavoidable.

We believe the half-injective correspondence between witness sets and cuts, as facilitated by the proportionate uncrossing theorem, is of independent interest, and will perhaps have other applications in this space of problems. We also emphasize here that the uncrossing theorem holds for all bipartite graphs, and not only regular bipartite graphs. Indeed, the graph G' on which we invoke this theorem does not inherit the regularity property of the original graph G. As another illustrative example, consider the celebrated Birkhoff-von Neumann theorem [18, 91] which says that every doubly stochastic matrix can be expressed as a convex combination of permutation matrices (i.e., perfect matchings). In some applications, it is of interest to do an iterative decomposition whereby a single matching is recovered in each iteration. The best-known bound for this problem, to our knowledge, is an $O(mb)$ time algorithm that follows from the work of Gabow and Kariv [33]; here b denotes the maximum number of bits needed to express any entry in M. The following theorem is an easy consequence of our proportionate uncrossing result.

Theorem 151 *Given an $n \times n$ doubly-stochastic matrix M with m non-zero entries, one can find a perfect matching in the support of M in $\tilde{O}(m + n^{1.5})$ expected time.*

The proof of this theorem and a discussion of known results about this problem are given in section 7.5. Though this result itself represents only a modest improvement over the earlier $O(mb)$ running time, it is an instructive illustration of the utility of the proportionate uncrossing theorem.

It is worth noting that while the analysis of Goel, Kapralov, and Khanna was along broadly similar lines (sample edges from the original graph, followed by running the Hopcroft-Karp algorithm), the proportionate uncrossing theorem developed in this paper requires significant new ideas and is crucial to incorporating the non-uniform sampling stage into our algorithm. Further, the running time of the Hopcroft-Karp algorithm is easily seen to be $\Omega(m\sqrt{n})$ even for the 2-regular

graph consisting of $\Theta(\sqrt{n})$ disjoint cycles of lengths $2, 4, \ldots, \sqrt{n}$ respectively; the stronger analysis for our special case requires both our uncrossing theorem as well as a stronger decomposition[1]. As a step in this analysis, we prove the independently interesting fact that after sampling edges from a d-regular bipartite graph with rate $\frac{c \ln n}{d}$, for some suitable constant c, we obtain a graph that has a matching of size $n - O(n/d)$ whp and such a matching can be found in $O(n/d)$ augmenting phases of the Hopcroft-Karp algorithm whp.

Organization: Section 7.1 reviews and presents some useful corollaries of relevant earlier work. In Section 7.2, we establish the proportionate uncrossing theorem. In section 7.3, we present and analyze our two-stage sampling scheme, and section 7.4 outlines the stronger analysis of the Hopcroft-Karp algorithm for our special case. Section 7.5 contains the proof of Theorem 151 and a discussion of known results on finding perfect matchings in the support of double stochastic matrices.

7.1 Preliminaries

In this section, we adapt and present the results of Chapter 2 as well as the Benczúr-Karger sampling theorem [16] for our purposes, and also prove a simple technical lemma for later use.

7.1.1 Bipartite Decompositions and Relevant Witness Pairs

Let $G = (P, Q, E)$ be a regular bipartite graph, with vertex set $P \cup Q$ and edge set $E \subseteq P \times Q$. Consider any partition of P into k sets $P_1, P_2, \ldots, P_k$, and a partition of Q into $Q_1, Q_2, \ldots, Q_k$. Let G_i denote the (not necessarily regular) bipartite graph (P_i, Q_i, E_i) where $E_i = E \cap (P_i \times Q_i)$. We will call this a "decomposition" of G.

Given $A \subseteq P$ and $B \subseteq Q$, define the witness set corresponding to the pair (A, B), denoted $W(A, B)$, as the set of all edges between A and $Q \setminus B$, and define the cut $C(A, B)$ as the set of all edges between $A \cup B$ and $(P \setminus A) \cup (Q \setminus B)$. The rest of the definitions in this section are with respect to some arbitrary but fixed decomposition of G.

Definition 152 *An edge $(u, v) \in E$ is relevant if $(u, v) \in E_i$ for some i.*

Definition 153 *Let E_R be the set of all relevant edges. A pair (A, B) is said to be relevant if*

1. *$A \subseteq P_i$ and $B \subseteq Q_i$ for some i,*
2. *$|A| > |B|$, and*
3. *There does not exist another $A' \in P_i$, $B' \in Q_i$, such that $A' \subset A$, $|A'| > |B'|$, and $W(A', B') \cap E_R \subseteq W(A, B) \cap E_R$.*

[1] It is known that the Hopcroft-Karp algorithm terminates quickly on bipartite expanders [71], but those techniques don't help in our setting since we start with an arbitrary regular bipartite graph.

Informally, a relevant pair is one which is contained completely within a single piece in the decomposition, and is "minimal" with respect to that piece. The following lemma is implicit in our results in Chapter 2 and is proved in section 7.6 for completeness.

Lemma 154 *Let $\mathcal{R}$ denote all relevant pairs (A, B) with respect to a decomposition of $G(P, Q, E)$, and let E_R denote all relevant edges. Consider any graph $G^* = (P, Q, E^*)$. If for all $(A, B) \in \mathcal{R}$, we have $W(A, B) \cap E^* \cap E_R \neq \phi$, then G^* has a perfect matching.*

7.1.2 A Corollary of Benczúr-Karger Sampling Scheme

The Benczúr-Karger sampling theorem [16] shows that for any graph, a relatively small *non-uniform* edge sampling rate suffices to ensure that every cut in the graph is hit by the sampled edges (i.e. it has a non-empty intersection) with high probability. The sampling rate used for each edge e inversely depends on its strength, as defined below.

Definition 155 *[16] A k-strong component of a graph H is a maximal vertex-induced subgraph of H with edge-connectivity k. The strength of an edge e in a graph H is the maximum value of k such that a k-strong component contains e.*

Definition 156 *Given a graph $H = (V, E)$, let $H_{[j]} = (V, E_{[j]})$ denote the subgraph of H restricted to edges of strength j or higher, where j is some integer in $\{1, 2, \ldots, |V|\}$.*

It is easy to see that whenever a cut in a graph $H(V, E)$ contains an edge of strength k, then the cut must contain at least k edges. Furthermore, for any $1 < j \leq |V|$, each connected component of graph $H_{[j]}$ is contained inside some connected component of $H_{[j-1]}$. The Benczúr-Karger theorem utilizes these properties to show that it suffices to sample each edge e with probability $\Theta(\min\{1, \ln n/s_e\})$.

We now extend this sampling result to any collection of edge-sets for which there exists an injection (one-one mapping) to cuts of comparable inverse strengths. The statement of our theorem 157 closely mirrors the Benczúr-Karger sampling theorem, and the proof is also along the same general lines. However, the proof does not follow from the Benczúr-Karger sampling theorem in a black-box fashion, so a proof is provided in section 7.7

Theorem 157 *Let $H(V, E)$ be any graph on n vertices, and let $\mathcal{C}$ denote the set of all possible edge cuts in H, and $\gamma \in (0, 1]$ be a constant. Let H' be a subgraph of H obtained by sampling each edge e in H with probability $p_e = \min\left\{1, \frac{c \ln n}{\gamma s_e}\right\}$, where s_e denotes the strength of edge e, and c is a suitably large constant. Further, let $\mathcal{X}$ be a collection of subset of edges, and let f be a one-one (not necessarily onto) mapping from $\mathcal{X}$ to $\mathcal{C}$ satisfying $\sum_{e \in X} 1/s_e > \gamma \sum_{e \in f(X)} 1/s_e$ for all $X \in \mathcal{X}$. Then*

$$\sum_{X \in \mathcal{X}} \Pr[\text{No edge in } X \text{ is chosen in } H'] \leq \frac{1}{n^2}.$$

The result below from [16] bounds the number of edges chosen by the sampling in Theorem 157.

Theorem 158 *Let $H(V, E)$ be any graph on n vertices, and let H' be a subgraph of H obtained by sampling each edge e in H with probability $p_e = \min\left\{1, \frac{c \ln n}{s_e}\right\}$, where s_e denotes the strength of edge e, and c is any constant. Then with probability at least $1 - \frac{1}{n^2}$, the graph H' contains at most $c'n \ln n$ edges, where c' is another suitably large constant.*

We conclude with a simple property of integer multisets that we will use later. A similar statement was used in [54] (lemma 4.5). A proof is provided in section 7.8 for completeness.

Lemma 159 *Let S_1 and S_2 be two arbitrary multisets of positive integers such that $|S_1| > \gamma|S_2|$ for some $\gamma > 0$. Then there exists an integer j such that*

$$\sum_{i \geq j \text{ and } i \in S_1} \frac{1}{i} > \gamma \left(\sum_{i \geq j \text{ and } i \in S_2} \frac{1}{i} \right).$$

7.2 Proportionate Uncrossing of Witness Sets

Consider a bipartite graph $G = (P, Q, E)$, with a non-negative weight function t defined on the edges. Assume further that we are given a set of "relevant edges" $E_R \subseteq E$. We can extend the definition of t to sets of edges, so that $t(S) = \sum_{e \in S} t(e)$, where $S \subseteq E$.

Definition 160 *For any $\gamma > 0$ and $A \subseteq P, B \subseteq Q$, the pair (A, B) is said to be γ-thick with respect to (G, t, E_R) if $t(W(A,B) \cap E_R) > \gamma t(C(A, B))$,* i.e., *the total weight of the relevant edges in $W(A, B)$ is* strictly *more than γ times the total weight of $C(A, B)$. A set of pairs $\mathcal{R} = \{(A_1, B_1), (A_2, B_2), \ldots, (A_K, B_K)\}$ where each $A_i \subseteq P$ and each $B_i \subseteq Q$ is said to be a γ-thick collection with respect to (G, t, E_R) if every pair $(A_i, B_i) \in \mathcal{R}$ is γ-thick.*

The quantities G, t, and E_R will be fixed for this section, and for brevity, we will omit the phrase "with respect to (G, t, E_R)" in the rest of this section.

Before defining proportionate uncrossings of witness sets, we will informally point out the motivation for doing so. If a pair (A, B) is γ-thick for some constant γ, and if we know that a sampling process where edge e is chosen with probability t chooses some edge from $C(A, B)$ with high probability, then increasing the sampling probability by a factor of $1/\gamma$ should result in some relevant edge from $W(A, B)$ being chosen with high probability as well, a fact that would be very useful in the rest of this paper. The sampling sub-routines that we employ in the rest of this paper are analyzed by using union-bound over all cuts, and in order to apply the same union bound, it would be useful if each witness set were to correspond to a unique cut. However, in figure 7.1(a), we show two pairs (A, B) and (X, Y) which are both $(1/2)$-thick but correspond to the same cut; we call this a "crossing" of the pairs (A, B) and (X, Y), drawing intuition from the figure. In general, we can have many witness sets that map to the same cut. We would like to "uncross" these witness sets by finding subsets of each witness set that map to unique cuts, but there is no way to uncross figure 7.1(a) in this fashion. Fortunately, and somewhat surprisingly, this is the worst case: any collection of γ-thick pairs can be uncrossed into another collection such that all the pairs in the new collection are also

γ-thick (hence the term proportionate uncrossing), every original witness set has a representative in this new collection, and no more than two new pairs have the same cut. Figure 7.1(b) shows two $\frac{1}{2}$-thick pairs that can be uncrossed using a single $\frac{1}{2}$-thick representative, $(A \cap X, B \cap Y)$. We will spend the rest of this section formalizing the notion of proportionate uncrossings and proving their existence. The uncrossing process is algorithmically inefficient, but we only need to demonstrate existence for the purpose of this paper. The arguments in this section represent the primary technical contribution of this paper; these arguments apply to bipartite graphs in general (not necessarily regular), and may be independently interesting.

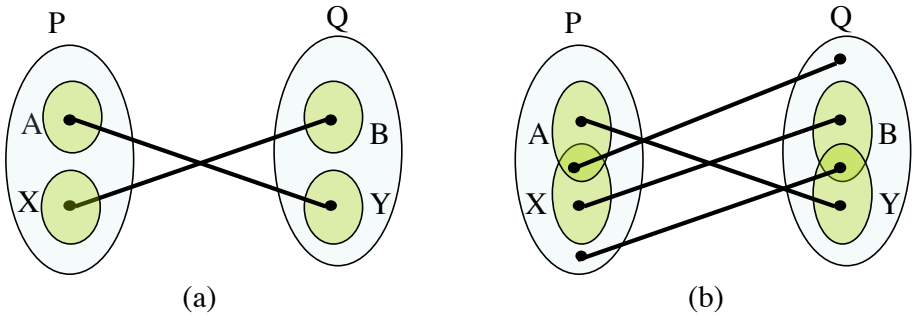


Figure 7.1: Both (a) and (b) depict two $\frac{1}{2}$-thick pairs (A, B) and (X, Y) that have different witness sets but the same cut (i.e. $W(A, B) \neq W(X, Y)$ but $C(A, B) = C(X, Y)$). The pairs in (a) can not be uncrossed, whereas the pairs in (b) can be uncrossed by choosing the single pair $(A \cap X, B \cap Y)$ as a representative.

7.2.1 Proportionate Uncrossings: Definitions and Properties

Definition 161 *A γ-uncrossing of a γ-thick collection $\mathcal{R}$ is another γ-thick collection of pairs $\mathcal{T}$ that satisfies the three properties below:*

P1: *For every pair $(A, B) \in \mathcal{R}$ there exists a pair $(A', B') \in \mathcal{T}$ such that $C(A', B') \subseteq C(A, B)$, and $W(A', B') \subseteq W(A, B)$. We will refer to (A', B') as a representative of (A, B).*

P2 *For every $(A', B') \in \mathcal{T}$, there exists $(A, B) \in \mathcal{R}$ such that $C(A', B') \subseteq C(A, B)$.*

P3: (Half-injectivity): *There can not be three distinct pairs $(A, B), (A', B')$, and (A'', B'') in $\mathcal{T}$ such that $C(A, B) = C(A', B') = C(A'', B'')$.*

Since $\mathcal{T}$ has the same (or larger) thickness as the thickness guarantee that we had for $\mathcal{R}$, it seems appropriate to refer to $\mathcal{T}$ as a proportionate uncrossing of $\mathcal{R}$.

Definition 162 *A γ-partial-uncrossing of a γ-thick collection $\mathcal{R}$ is another γ-thick collection of pairs $\mathcal{T}$ which satisfies properties P1,P2 above but not necessarily P3.*

The following three lemmas follow immediately from the two definitions above, and it will be useful to state them explicitly. Informally, the first says that every collection is its own partial uncrossing,

the second says that uncrossings can be composed, and the third says that the union of the partial uncrossings of two collections is a partial uncrossing of the union of the collections.

Lemma 163 *If $\mathcal{R}$ is a γ-thick collection, then $\mathcal{R}$ is a γ-partial uncrossing of itself.*

Lemma 164 *If $\mathcal{S}$ is a γ-partial uncrossing of a γ-thick collection $\mathcal{R}$, and $\mathcal{T}$ is a γ-uncrossing of $\mathcal{S}$, then $\mathcal{T}$ is also a γ-uncrossing of $\mathcal{R}$.*

Lemma 165 *If $\mathcal{R}_1$ and $\mathcal{R}_2$ are two γ-thick connections, $\mathcal{T}_1$ is a γ-partial-uncrossing of $\mathcal{R}_1$, and $\mathcal{T}_2$ is a γ-partial-uncrossing of $\mathcal{R}_2$, then $\mathcal{T}_1 \cup \mathcal{T}_2$ is a γ-partial-uncrossing of $\mathcal{R}_1 \cup \mathcal{R}_2$.*

7.2.2 Proportionate Uncrossings: An Existence Theorem

The main technical result of this section is the following:

Theorem 166 *For every γ-thick collection $\mathcal{R}$, there exists a γ-uncrossing of $\mathcal{R}$.*

The proof is via induction over the "largest cut" corresponding to any pair in the collection $\mathcal{R}$; each inductive step "uncrosses" the witness sets which corresponds to this largest cut. Before proving this theorem, we need to provide several useful definitions and also establish a key lemma.

Define some total ordering $\prec$ over all subsets of E which respects set cardinality, so that if $|E_1| < |E_2|$ then $E_1 \prec E_2$. Overload notation to use $C(\mathcal{R})$ to denote the set of cuts $\{C(A,B) : (A,B) \in \mathcal{R}\}$. Analogously, use $W(\mathcal{R})$ to denote the set of witness sets corresponding to pairs in $\mathcal{R}$. Since $C(A,B)$ may be equal to $C(A',B')$ for $(A,B) \neq (A',B')$, it is possible that $|C(\mathcal{R})|$ may be smaller than $|\mathcal{R}|$. In fact, if $\mathcal{R}$ and $|C(\mathcal{R})|$ are equal, then $\mathcal{R}$ is its own γ-uncrossing and the theorem is trivially true. Similarly, it is possible that $W(A,B)$ is equal to $W(A',B')$ for two different pairs (A,B) and (A',B') in $\mathcal{R}$. However, suppose $W(A,B) = W(A',B')$ and $C(A,B) = C(A',B')$ for two different pairs (A,B) and (A',B') in $\mathcal{R}$. In this case, we can remove one of the two pairs from the collection to obtain a new collection $\mathcal{R}'$; it is easy to see that a γ-uncrossing of $\mathcal{R}'$ is also a γ-uncrossing of $\mathcal{R}$. So we will assume without loss of generality that for any two pairs (A,B) and (A',B') in $\mathcal{R}$, either $W(A,B) \neq W(A',B')$ or $C(A,B) \neq C(A',B')$; we will call this the *non-redundancy* assumption.

We will now prove a key lemma which contains the meat of the uncrossing argument. When we use this lemma later in the proof of theorem 166, we will only use the fact that there exists a γ-partial-uncrossing of $\mathcal{R}$, where $\mathcal{R}$ satisfies the preconditions of the lemma. However, the stronger claim of existence of a γ-uncrossing does not require much additional work and appears to be an interesting graph theoretic argument in its own right, so we prove this stronger claim.

Lemma 167 *If $\mathcal{R}$ is a γ-thick collection such that $|\mathcal{R}| > 2$, $\mathcal{R}$ satisfies the non-redundancy assumption, and $C(\mathcal{R})$ contains a single set S, then there exists a γ-uncrossing $\mathcal{T}$ of $\mathcal{R}$. Further, for every pair $(A,B) \in \mathcal{T}$, we have $C(A,B) \subset S$.*

Proof: Let $\mathcal{R} = \{(A_1,B_1),(A_2,B_2),\ldots,(A_J,B_J)\}$. Since $C(A_i,B_i) = S$ for all i, we know by the non-redundancy assumption that $W(A_i,B_i) \neq W(A_{i'},B_{i'})$ for $i \neq i'$. We break the proof down into multiple stages.

1. *Definition of Venn witnesses and Venn cuts.* For any J-dimensional bit-vector $b \in \{0,1\}^J$, define

$$A_{(b)} = \left(P \cap \left(\bigcap_{b_i=1} A_i\right)\right) \setminus \left(\bigcup_{b_i=0} A_i\right),$$

and similarly,

$$B_{(b)} = \left(Q \cap \left(\bigcap_{b_i=1} B_i\right)\right) \setminus \left(\bigcup_{b_i=0} B_i\right).$$

We overload notation and use $W_{(b)}$ to denote the witness set $W(A_{(b)}, B_{(b)})$ and $C_{(b)}$ to denote the cut set $C(A_{(b)}, B_{(b)})$. A node u belongs to $A_{(b)}$ if it is in every set A_i such that $b_i = 1$ and not in any of the sets A_i for which $b_i = 0$. Thus, each $A_{(b)}$ corresponds to one of the regions in the Venn diagram of the sets $A_1, A_2, \ldots, A_J$, and the analogous statement holds for each $B_{(b)}$. Hence, we will refer to the sets $W_{(b)}$ and $C_{(b)}$ as the Venn-witness and the Venn-cut for b, respectively, and refer to the pair $(A_{(b)}, B_{(b)})$ as a Venn pair. Also, we will use $\overline{b}$ to refer to a vector which differs from b in every bit.

2. *The special structure of Venn witnesses and Venn cuts.* Consider an edge (u, v) that goes out of $A_{(b)}$. Suppose that edge goes to $B_{(d)}$ where $d \neq b$ and $d \neq \overline{b}$. Then there must exist $1 \leq i, i' \leq J$ such that $b_i = d_i$ and $b_{i'} \neq d_{i'}$. Since $b_i = d_i$, either $u \in A_i, v \in B_i$ (if $b_i = d_i = 1$) or $u \notin A_i, v \notin B_i$ (if $b_i = d_i = 0$). In either case the edge (u, v) does not belong to the cut $C(A_i, B_i)$, and since all pairs in $\mathcal{R}$ have the same cut S, we conclude that $(u, v) \notin S$. On the other hand, since $b_{i'} \neq d_{i'}$, either $u \in A_{i'}, v \notin B_{i'}$ (if $b_{i'} = 1, d_{i'} = 0$) or $u \notin A_{i'}, v \in B_{i'}$ (if $b_{i'} = 0, d_{i'} = 1$). In either case the edge (u, v) belongs to the cut $C(A_{i'}, B_{i'})$ and hence to S, which is a contradiction. Thus, *any edge from $A_{(b)}$ goes to either $B_{(b)}$ or $B_{(\overline{b})}$.*

If the edge (u, v) goes to $B_{(b)}$ then it does not belong to any witness set in $W(\mathcal{R})$, any Venn witness set, any Venn cut, or S. If (u, v) goes to $B_{(\overline{b})}$ then it belongs to S, to the Venn witness set $W_{(b)}$, to the Venn cuts $C_{(b)}$ and $C_{(\overline{b})}$, and to no other Venn witness set or Venn cut. This edge also belongs to $W(A_i, B_i)$ for all i such that $b_i = 1$. These observations, and the definitions of Venn witnesses, cuts, and pairs easily lead to the following consequences:

$$W_{(b)} \cap W_{(d)} = \emptyset \text{ if } b \neq d, \tag{7.1}$$

$$W(A_i, B_i) = \bigcup_{b \in \{0,1\}^J : b_i = 1} W_{(b)}, \tag{7.2}$$

$$C_{(b)} = C_{(\overline{b})}, \tag{7.3}$$

$$C_{(b)} \cap C_{(d)} = \emptyset \text{ if } b \neq d \text{ and } b \neq \overline{d}, \tag{7.4}$$

$$(\forall i, 1 \leq i \leq J) : S = \bigcup_{b \in \{0,1\}^J : b_i = 1} C_{(b)}, \tag{7.5}$$

and finally,

$$W_{(b)} \cup W_{(\bar{b})} = C_{(b)}. \tag{7.6}$$

3. *The collection $\mathcal{T}$.* Define $\mathcal{T}$ to consist of all γ-thick Venn pairs $(A_{(b)}, B_{(b)})$ where b is not the all zero vector.

4. *Proving that $\mathcal{T}$ is a γ-uncrossing of $\mathcal{R}$.* **(P1):** Fix some $i, 1 \leq i \leq J$. Since $\mathcal{R}$ is a γ-thick collection, it follows from the definition that (A_i, B_i) must be a γ-thick pair. From equations 7.2 and 7.1, we know that $t(W(A_i, B_i) \cap E_R) = \sum_{b \in \{0,1\}^J : b_i = 1} t(W_{(b)} \cap E_R)$. We also know, from equations 7.4 and 7.5, that $t(S) = \sum_{b \in \{0,1\}^J : b_i = 1} t(C_{(b)})$. Hence, there must be some $b \in \{0,1\}^J$ such that $b_i = 1$ and $(A_{(b)}, B_{(b)})$ is γ-thick, which in turn implies that $(A_{(b)}, B_{(b)})$ is in $\mathcal{T}$. This is the representative of (A_i, B_i) and hence $\mathcal{T}$ satisfies P1. **(P2):** This follows trivially from equation 7.5. **(P3):** From equation 7.4 we know that there are only two possible Venn pairs (specifically, $(A_{(b)}, B_{(b)})$ and $(A_{(\bar{b})}, B_{(\bar{b})})$) that have the same non-empty cut $C_{(b)}$. Observe that our definition of γ-thickness involves "strict inequality", and hence Venn pairs where the Venn witness set and the Venn cut are both empty can't be γ-thick and can't be in $\mathcal{T}$.

5. *Proving that $C(X, Y) \subset S$ for all pairs $(X, Y) \in \mathcal{T}$.* Any cut $C(A, B) \in C(\mathcal{T})$ is of the form $C_{(b)}$ for some J-dimensional bit vector b. Each $C_{(b)} \subseteq S$, from equation 7.5. We will now show that this containment is strict. Suppose not, *i.e.*, there exists some $C_{(b)} = S$. By equation 7.3, $C_{(\bar{b})} = S$ as well. Since $J > 2$, either b or $\bar{b}$ must have two bits that are set to 1; without loss of generality, assume that $b_1 = b_2 = 1$. From equations 7.1 and 7.6, we know that $C_{(b)}$ (and hence S) is the disjoint union of $W_{(b)}$ and $W_{(\bar{b})}$. Any edge in $W_{(b)}$ must belong to both $W(A_1, B_1)$ and $W(A_2, B_2)$, whereas any edge in $W_{(\bar{b})}$ can not belong to either $W(A_1, B_1)$ or $W(A_2, B_2)$. Hence, $W(A_1, B_1) = W(A_2, B_2) = W_{(b)}$ which contradicts the non-redundancy assumption on $\mathcal{R}$. Therefore, we must have $C_{(b)} \subset S$. ■

Proof of Theorem 166: The proof will be by induction over the largest set in $C(\mathcal{R})$ according to the ordering $\prec$. Let $M(\mathcal{R})$ denote this largest set.

For the base case, suppose $M(\mathcal{R})$ is the smallest set S under the ordering $\prec$. Then S must be singleton, $C(\mathcal{R})$ must have just a single set S, and $W(\mathcal{R})$ must also have a single witness set, which must be the same as S since $\mathcal{R}$ is γ-thick. By the non-redundancy assumption, $\mathcal{R}$ must have at most one pair, and is its own γ-uncrossing.

For the inductive step, consider any possible cut S and assume that the theorem is true when $M(\mathcal{R}) \prec S$. We will show that the theorem is also true when $M(\mathcal{R}) = S$, which will complete the inductive proof.

Suppose there is a unique $(A, B) \in \mathcal{R}$ such that $C(A, B) = S$. Intuitively, one would expect this to be the easy case, since there is no "uncrossing" to be done for S, and indeed, this case is quite straightforward. Define $\mathcal{R}' = \mathcal{R} - (A, B)$. Let $\mathcal{T}'$ denote a γ-uncrossing of $\mathcal{R}$, which is guaranteed to exist by the inductive hypothesis. Since $\mathcal{T}'$ is γ-thick, so is $\mathcal{T} = \mathcal{T}' \cup \{(A, B)\}$. The pair (A, B) clearly has a representative in $\mathcal{T}$ (itself), and any $(A', B') \in \mathcal{R} - (A, B)$ has a representative in $\mathcal{T}'$ and hence

also in $\mathcal{T}$. Thus, $\mathcal{T}$ satisfies property P1 for being a γ-uncrossing of $\mathcal{R}$. Every set in $C(\mathcal{T}')$ is a subset of some cut in $C(\mathcal{R}')$ (by property P2) and $C(A,B)$ is also in $C(\mathcal{R})$, and hence $\mathcal{T}$ satisfies property P2 for being a γ-uncrossing of $\mathcal{R}$. Every set in $\mathcal{T}'$ is smaller than $C(A,B)$ according to $\prec$ and $\mathcal{T}'$ satisfies property P3. Hence, $\mathcal{T}$ also satisfies property P3. Thus, $\mathcal{T}$ is a γ-uncrossing of $\mathcal{R}$. If there are exactly two distinct pairs (A,B) and (A',B') in $\mathcal{R}$ such that $C(A,B) = C(A',B') = S$, then the same argument works again, except that $\mathcal{R}' = \mathcal{R} \setminus \{(A,B),(A',B')\}$ and $\mathcal{T} = \mathcal{T}' \cup \{(A,B),(A',B')\}$.

We now need to tackle the most interesting case of the inductive step, where there are more than two pairs in $\mathcal{R}$ that correspond to the same cut S. Write $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ where $C(A,B) \prec S$ for all $(A,B) \in \mathcal{R}_1$ and $C(A,B) = S$ for all $(A,B) \in \mathcal{R}_2$. Recall that for two different pairs (A,B) and (A',B') in $\mathcal{R}_2$, we must have $W(A,B) \neq W(A',B')$ by the non-redundancy assumption. From lemma 167, there exists a γ-partial-uncrossing, say $\mathcal{S}_2$, of $\mathcal{R}_2$ with the property that for every set $S' \in C(\mathcal{S}_2)$, we have $S' \subset S$, and hence $S' \prec S$. By lemma 163, we know that $\mathcal{R}_1$ is its own γ-partial-uncrossing. Further, by definition of $\mathcal{R}_1$, every set $S' \in C(\mathcal{R}_1)$ must satisfy $S' \prec S$. Define $\mathcal{S} = \mathcal{R}_1 \cup \mathcal{S}_2$. By lemma 165, $\mathcal{S}$ is a γ-partial-uncrossing of $\mathcal{R}_1 \cup \mathcal{R}_2$, *i.e.*, of $\mathcal{R}$. Further, for every cut $S' \in C(\mathcal{S})$, we have $S' \prec S$. Hence, by our inductive hypothesis, there exists a γ-uncrossing of $\mathcal{S}$; let $\mathcal{T}$ be a γ-uncrossing of $\mathcal{S}$. By lemma 164, $\mathcal{T}$ is also a γ-uncrossing of $\mathcal{R}$, which completes the inductive proof. ■

Remark 168 *An alternate approach to relating cuts and witness sets is to suitably modify the proof of the Benczúr-Karger sampling theorem, circumventing the need for the proportionate uncrossing theorem. The idea is based on the observation that Karger's sampling theorem also holds for vertex cuts in graphs. Since Benczúr-Karger sampling theorem is proved using multiple invocations of Karger's sampling theorem, it is possible to set up a correspondence between cuts and witness sets using a vertex-cut version of the Benczúr-Karger sampling theorem. However, we prefer to use here the approach based on the proportionate uncrossing theorem as it is an interesting combinatorial statement in its own right.*

7.3 An $\tilde{O}(n^{1.5})$ Time Algorithm for Finding a Perfect Matching

We present here an $\tilde{O}(n^{1.5})$ time randomized algorithm to find a perfect matching in a given d-regular bipartite graph $G(P,Q,E)$ on $2n$ vertices. Throughout this section, we follow the convention that for any pair (A,B), the sets $C(A,B)$ and $W(A,B)$ are defined with respect to the graph G. Our starting point is the following theorem, which we established in Chapter 2 [2]

Theorem 169 *Let $G(P,Q,E)$ be a d-regular bipartite graph, ϵ any number in $(0,\frac{1}{2})$, and c a suitably large constant that depends on ϵ. There* exists *a decomposition of G into $k = O(n/d)$ vertex-disjoint bipartite graphs, say $G_1 = (P_1,Q_1,E_1), G_2 = (P_2,Q_2,E_2), \ldots, G_k = (P_k,Q_k,E_k)$, such that*

[2] Part 1 of theorem 169 corresponds to theorem 4 in Chapter 2, part 2 is proved as part of the proof of theorem 1, and part 3 combines remark 6 with Karger's sampling theorem [52].

1. *Each G_i contains at least $d/2$ perfect matchings, and the minimum cut in each G_i is $\Omega(d^2/n)$.*

2. *Let $\mathcal{R}$ denote the set of relevant pairs with respect to this decomposition, and E_R denote the set of relevant edges. Then for each (A, B) in $\mathcal{R}$, we have $|W(A, B) \cap E_R| \geq \frac{1}{2}|C(A, B)|$.*

3. *Let $G'(P, Q, E')$ be a random graph generated by sampling the edges of G* uniformly *at random with probability $p = \frac{cn \ln n}{d^2}$. Then with probability at least $1 - 1/n$, for every pair $(A, B) \in \mathcal{R}$,*

$$|W(A, B) \cap E' \cap E_R| > (1 - \epsilon)p|W(A, B) \cap E_R| > \left(\frac{1 - \epsilon}{2(1 + \epsilon)}\right)|C(A, B) \cap E'|.$$

The last condition above says that in addition to all cuts, all relevant witness edge sets are also preserved to within $(1 \pm \epsilon)$ of their expected value in G', with high probability. We emphasize here that the decomposition highlighted in Theorem 169 will be used only in the analysis of our algorithm; the algorithm itself is oblivious to this decomposition.

Our algorithm consists of the following three steps.

(S1) Generate a random graph $G' = (P, Q, E')$ by sampling edges of G *uniformly* at random with probability $p = \frac{c_1 n \ln n}{d^2}$ where c_1 is a constant as in Theorem 169[3]. We choose ϵ to be any fixed constant not larger than 0.2.

(S2) The graph G' contains $O(\frac{n^2 \ln n}{d})$ edges w.h.p. We now run the Benczúr-Karger sampling algorithm [16] that takes $O(|E'| \ln^2 n)$ time to compute the strength s_e of every edge e, and samples each edge e with probability p_e;[4] here p_e is as given by Theorem 157 with $\gamma = 1/3$. We show below that w.h.p. the graph $G'' = (P, Q, E'')$ obtained from this sampling contains a perfect matching.

(S3) Finally, we run the Hopcroft-Karp algorithm to obtain a maximum cardinality matching in G'' in $O(n^{1.5} \ln n)$ time since by Theorem 158, G'' contains $O(n \ln n)$ edges w.h.p.

Running time: With high probability, the running time of this algorithm is bounded by $O(\frac{n^2}{d} \ln^3 n + n^{1.5} \ln n)$. Since we can always use the algorithm of Cole, Ost, and Schirra [23] instead, the final running time is $O(\min\{m, \frac{n^2}{d} \ln^3 n + n^{1.5} \ln n\})$. This reduces to $O(m)$ if $d \leq \sqrt{n} \ln n$; to $O(n^{1.5} \ln n)$ when $d \geq \sqrt{n} \ln^2 n$; and to at most $O((n \ln n)^{1.5})$ in the narrow range $\sqrt{n} \ln n < d < \sqrt{n} \ln^2 n$.

Correctness: To prove correctness, we need to show that G'' contains a perfect matching w.h.p.

Theorem 170 *The graph G'' contains a perfect matching with probability $1 - O(1/n)$.*

Proof: Consider the decomposition defined in Theorem 169. Let $\mathcal{R}$ denote the set of relevant pairs with respect to this decomposition, and let E_R denote the set of all relevant edges with respect to

[3]The time required for this sampling is proportional to the number of edges chosen, assuming the graph is presented in an adjacency list representation with each list stored in an array.

[4]In fact, this sampling algorithm computes an upper bound on s_e, but this only affects the running time and the number of edges sampled by a constant factor.

this decomposition. We will now focus on proving that, with high probability, for every $(A, B) \in \mathcal{R}$, $W(A, B) \cap E_R \cap E'' \neq \emptyset$; by Lemma 154, this is sufficient to prove the theorem.

For convenience, define $W'(A, B) = W(A, B) \cap E'$ and $C'(A, B) = C(A, B) \cap E'$. Assume for now that the low-probability event in Theorem 169 does not occur. Thus, by choosing $\epsilon \leq 0.2$, we know that for $\gamma = 1/3$, every relevant pair $(A, B) \in \mathcal{R}$ satisfies $|W'(A, B) \cap E_R| > \gamma |C'(A, B)|$.

Let s'_e denote the strength of e in G'. Recall that $G'_{[j]} = (V, E'_{[j]})$ is the graph with the same vertex set as G' but consisting of only those edges in E' which have strength at least j. Define $W'_{[j]}(A, B)$ to be the set of all edges in $W'(A, B) \cap E'_{[j]}$; define $C'_{[j]}(A, B)$ analogously. Define $t(e) = 1/s'_e$. Since $|W'(A, B) \cap E_R| > \gamma |C'(A, B)|$, by Lemma 159, there must exist a j such that

$$\sum_{e \in (W'(A,B) \cap E_R), s'_e \geq j} \frac{1}{s'_e} > \gamma \sum_{e \in C'(A,B), s'_e \geq j} \frac{1}{s'_e} > 0,$$

which implies that (A, B) is γ-thick with respect to $(G'_{[j]}, t, E_R)$, as defined in Definition 160. Partition $\mathcal{R}$ into $\mathcal{R}_{[1]}$, $\mathcal{R}_{[2]}, \ldots, \mathcal{R}_{[n]}$, such that if $(A, B) \in \mathcal{R}_{[j]}$ then (A, B) is γ-thick with respect to $(G'_{[j]}, t, E_R)$, breaking ties arbitrarily if (A, B) can belong to multiple $\mathcal{R}_{[j]}$. Consider an arbitrary non-empty $\mathcal{R}_{[j]}$. Let $\mathcal{T}$ represent a γ-uncrossing of $\mathcal{R}_{[j]}$, as guaranteed by Theorem 166. By property P3, no three pairs in a γ-uncrossing can have the same cut; partition $\mathcal{T}$ into $\mathcal{T}_1$ and $\mathcal{T}_2$ such that every pair $(A, B) \in \mathcal{T}_1$ has a unique cut $C'_{[j]}(A, B)$ and the same holds for $\mathcal{T}_2$. We focus on $\mathcal{T}_1$ for now. For any $(A, B) \in \mathcal{T}_1$, define $Y(A, B) = W'_{[j]}(A, B) \cap E_R$. Define $\mathcal{X} = \{Y(A, B) : (A, B) \in \mathcal{T}_1\}$. For any $X \in \mathcal{X}$, define $f(X) = C'_{[j]}(A, B)$ for some arbitrary $(A, B) \in \mathcal{T}_1$ such that $X = Y(A, B)$. The function f is one-one by construction, and since (A, B) is γ-thick, we know that $\sum_{e \in X} 1/s'_e > \gamma \sum_{e \in f(X)} 1/s'_e$. Thus, $\mathcal{X}$ satisfies the preconditions of Theorem 157. Further, the sampling probability p_e in step **(S2)** of the algorithm is chosen to correspond to $\gamma = 1/3$. Thus, with probability at least $1 - 1/n^2$, $X \cap E''$ is non-empty for all $X \in \mathcal{X}$, *i.e.*, $W'_{[j]}(A, B) \cap E_R \cap E'' \neq \emptyset$ for all $(A, B) \in \mathcal{T}_1$. Since $G'_{[j]}$ is a subgraph of G', we can conclude that $W'(A, B) \cap E_R \cap E'' \neq \emptyset$ for all $(A, B) \in \mathcal{T}_1$ with probability at least $1 - 1/n^2$.

Since the analogous argument holds for $\mathcal{T}_2$, we obtain $W'(A, B) \cap E_R \cap E'' \neq \emptyset$ for all $(A, B) \in \mathcal{T}$ with probability at least $1 - 2/n^2$. Since $\mathcal{T}$ is a γ-uncrossing of $\mathcal{R}_{[j]}$, we use property P1 to conclude that $W'(A, B) \cap E_R \cap E'' \neq \emptyset$ for all $(A, B) \in \mathcal{R}_{[j]}$, again with probability at least $1 - 2/n^2$. Applying the union bound over all j, we further conclude that $W'(A, B) \cap E_R \cap E'' \neq \emptyset$ for all $(A, B) \in \mathcal{R}$ with probability at least $1 - 2/n$. As mentioned before, this suffices to prove that G'' has a perfect matching with probability at least $1 - 2/n$, by Lemma 154. We assumed that condition 3 in theorem 169 is satisfied; this is violated with probability at most $\frac{1}{n}$, which proves that G'' has a perfect matching with probability at least $1 - \frac{3}{n}$. ■

As presented above, the algorithm takes time $\min\{\tilde{O}(n^{1.5}), O(m)\}$ with high probability, and outputs a perfect matching with probability $1 - O(1/n)$. We conclude with two simple observations. First. it is easy to convert this into a Monte Carlo algorithm with a worst case running-time of $\min\{\tilde{O}(n^{1.5}), O(m)\}$, or a Las Vegas algorithm with an expected running-time of $\min\{\tilde{O}(n^{1.5}), O(m)\}$. If either the sampling process in steps **(S1)** or **(S2)** returns too many edges, or step **(S3)** does not produce a perfect matching, then (a) abort the computation to get a Monte Carlo algorithm, or (b)

run the $O(m)$ time algorithm of Cole, Ost, and Schirra [23] to get a Las Vegas algorithm. Second, by choosing larger constants during steps **(S1)** and **(S2)**, it is easy to amplify the success probability to be at least $1 - O(\frac{1}{n^j})$ for any fixed $j \geq 1$.

7.4 An Improved $O\left(\min\{nd, (n^2 \ln^3 n)/d\}\right)$ Bound on the Runtime

In this section we give an improved analysis of the runtime of the Hopcroft-Karp algorithm on the subsampled graph, ultimately leading to a bound of $O\left(\min\{nd, (n^2 \ln^3 n)/d\}\right)$ for our algorithm. The main ingredients of our analysis are (1) a decomposition of the graph G into $O(n/d)$ vertex-disjoint $\Omega(d)$-edge-connected subgraphs, (2) a modification of the uncrossing argument that reveals properties of sufficiently unbalanced witness sets in the sampled graph obtained in step **S2**, and (3) an upper bound on length of the shortest augmentating path in the sampled graph relative to any matching of size smaller than $n - 2n/d$.

7.4.1 Combinatorial uncrossings

Theorem 172 below, which we state for general bipartite graphs, requires a variant of the uncrossing theorem that we formulate now. We introduce the definition of combinatorial uncrossings:

Definition 171 *Let $\mathcal{R}$ be any collection of pairs $(A, B), A \subseteq P, B \subseteq Q$. A combinatorial uncrossing of $\mathcal{R}$ is a tuple $(\mathcal{T}, \mathcal{I})$, where $\mathcal{T}$ is another collection and $\mathcal{I}$ is a mapping from $\mathcal{R}$ to subsets of $\mathcal{T}$, such that the following properties are satisfied:*

Q1: *For all $(A, B) \in \mathcal{R}$*

1. *$\{W(A', B')\}_{(A',B') \in \mathcal{I}(A,B)}$ are disjoint;*
2. *$\{C(A', B')\}_{(A',B') \in \mathcal{I}(A,B)}$ are disjoint;*
3. *$\{A' \cup B'\}_{(A',B') \in \mathcal{I}(A,B)}$ are disjoint;*
4. *$A' \subseteq A, B' \subseteq B$ for all $(A', B') \in \mathcal{I}(A, B)$;*
5.

$$W(A, B) = \bigcup_{(A',B') \in \mathcal{I}(A,B)} W(A', B')$$
$$C(A, B) = \bigcup_{(A',B') \in \mathcal{I}(A,B)} C(A', B').$$

Q2: *(Half-injectivity) There cannot be three distinct pairs $(A, B), (A', B'), (A'', B'')$ in $\mathcal{T}$ such that $C(A, B) = C(A', B') = C(A'', B'')$.*

The proof of existence of combinatorial uncrossings is along the lines of the proof of existence of γ-thick uncrossings, so we omit it here.

For a graph H we denote $W_H(A,B) = W(A,B) \cap E(H)$ and $C_H(A,B) = C(A,B) \cap E(H)$, and omit the subscript when the underlying graph is fixed.

Theorem 172 *Let G^* be a graph obtained by sampling edges uniformly at random with probability p from a bipartite graph $G = (P, Q, E)$ on $2n$ vertices with a minimum cut of size κ. Then there exists a constant $c > 0$ such that for all $\epsilon > 0$ if $p > \frac{c \ln n}{\epsilon^2 \kappa}$ then w.h.p. for all $A \subseteq P$, and $B \subseteq Q$, we have*

$$p|W_G(A,B)| - \epsilon p|C_G(A,B)| \leq |W_{G^*}(A,B)| \leq p|W_G(A,B)| + \epsilon p|C_G(A,B)|.$$

Proof: Define $\mathcal{R}$ as the set of pairs $(A,B), A \subseteq P \cap V(G), B \subseteq Q \cap V(G)$. Denote a combinatorial uncrossing of $\mathcal{R}$ by $(\mathcal{T}, \mathcal{I})$. We first prove the statement for pairs from $\mathcal{T}$, and then extend it to pairs from $\mathcal{R}$ to obtain the desired result.

Consider a pair $(A,B) \in \mathcal{T}$. Denote $\Delta_G(A,B) = |W_{G^*}(A,B)| - p|W_G(A,B)|$. We shall write $W(A,B)$ and $C(A,B)$ instead of $W_G(A,B)$ and $C_G(A,B)$ in what follows for brevity. We have by Chernoff bounds that for a given pair $(A,B) \in \mathcal{T}$

$$\begin{aligned}\mathbf{Pr}\left[|\Delta_G(A,B)| > \epsilon p|C(A,B)|\right] &< \exp\left[-\left(\frac{\epsilon|C(A,B)|}{|W(A,B)|}\right)^2 \frac{p|W(A,B)|}{2}\right] \\ &\leq \exp\left[-\epsilon^2\left(\frac{p|C(A,B)|}{2}\right)\right]\end{aligned}$$

since $|C(A,B)| \geq |W(A,B)|$. Since $\mathcal{T}$ satisfies Q2, we get that

$$\begin{aligned}&\mathbf{Pr}\left[\exists (A,B) \in \mathcal{T} : |\Delta_G(A,B)| > \epsilon p|C(A,B)|\right] \\ &< \sum_{W(A,B) \in W(\mathcal{T})} \exp\left[-\epsilon^2 p|C(A,B)|/2\right] \leq 2 \sum_{C(A,B) \in C(\mathcal{T})} \exp\left[-\epsilon^2 p|C(A,B)|/2\right] = O(n^{-r})\end{aligned}$$

for $c = 2(r+2)$ by Corollary 2.4 in [52]. This implies that for $c \geq 2(r+2)$ we have with probability $1 - O(n^{-r})$ for all $(A,B) \in \mathcal{T}$

$$|\Delta_G(A,B)| \leq \epsilon p|C(A,B)|. \tag{7.7}$$

Now consider any pair $(A,B) \in \mathcal{R}$. Summing (7.7) over all $(A',B') \in \mathcal{I}(A,B)$ and using properties Q1.1-5, we get

$$|\Delta_G(A,B)| \leq \sum_{(A',B') \in \mathcal{I}(A,B)} \epsilon p|C(A',B')| = \epsilon p|C(A,B)|,$$

for all $(A,B) \in \mathcal{R}$ as required. ■

7.4.2 Decomposition of the graph G

Corollary 176, which relates the size of sufficiently unbalanced witness sets in the sampled graph to the size of the corresponding cuts is the main result of this subsection. It follows from theorem 172

and a stronger decomposition of bipartite d-regular graphs that we outline now.

Theorem 173 *Any d-regular graph G with $2n$ vertices can be decomposed into vertex-disjoint induced subgraphs $G_1 = (P_1, Q_1, E_1), G_2 = (P_2, Q_2, E_2), ..., G_k = (P_k, Q_k, E_k)$, where $k \leq 4n/d + 1$, that satisfy the following properties:*

1. *The minimum cut in each G_i is at least $d/8$.*
2. *$\sum_{i=1}^{k+1} |\delta_G(V(G_i))| \leq 2n$.*

To prove Theorem 173, we give a procedure that decomposes the graph G into vertex-disjoint induced subgraphs $G_1(P_1, Q_1, E_1)$, $G_2(P_2, Q_2, E_2), \ldots, G_k(P_k, Q_k, E_k)$, $k \leq 4n/d + 1$ such that the min-cut in G_j is at least $d/8$ and at most n edges run between pieces of the decomposition.

The procedure is as follows. Initialize $H_1 := G$, and set $i := 1$.

1. Find a smallest proper subset $X_i \subset V(H_i)$ such that $|\delta_{H_i}(X_i)| < d/4$. If no such set exists, define G_i to be the graph H_i and terminate.
2. Define G_i to be the subgraph of H_i induced by vertices in X_i, i.e. $X_i = P_i \cup Q_i = V(G_i)$. Also, define H_{i+1} to be the graph H_i with vertices from X_i removed.
3. Increment i and go to step 1.

We now prove that the output of the decomposition procedure satisfies the properties claimed above.

Lemma 174 *The min-cut in G_i is greater than $d/8$.*

Proof: If G_i contains a single vertex the min-cut is infinite by definition, so we assume wlog that G_i contains at least two vertices. The proof is essentially the same as the proof of property **P1** of the decomposition procedure in Chapter 2.

Suppose that there exists a cut (V, V^c) in G_i where $V \subset V(G_i)$ and $V^c = V(G_i) \setminus V$, such that $|\delta_{G_i}(V)| \leq d/8$ (note that it is possible that $V \cap P_i \neq \emptyset$ and $V \cap Q_i \neq \emptyset$). We have $|\delta_{H_i}(V) \setminus \delta_{G_i}(V)| + |\delta_{H_i}(V^c) \setminus \delta_{G_i}(V^c)| < d/4$ by the choice of X_i in (1). Suppose without loss of generality that $|\delta_{H_i}(V) \setminus \delta_{G_i}(V)| < d/8$. Then $|\delta_{H_i}(V)| < d/4$ and $V \subset X_i$, which contradicts the choice of X_i as the smallest cut of value at most $d/4$ in step (1) of the procedure. ■

Lemma 175 *The number of steps in the decomposition procedure is $k \leq 4n/d$, and at most n edges are removed in the process.*

Proof: We call a vertex $v \in V(G_i)$ *bad* if its degree in G_i is smaller than $d/2$. Note that for each $1 \leq i \leq k$ either G_i contains a bad vertex or $V(G_i) \geq d$.

Note that since strictly fewer than $d/4$ edges are removed in each iteration, the number of bad vertices created in the first j iterations is strictly less than $j(d/4)/(d/2) = j/2$. Hence, during at least half of the j iterations at least d vertices were removed from the graph, i.e.

$$\sum_{i=1}^{j} |V(G_i)| \geq (j/2) \cdot d = jd/2.$$

This implies that the process terminates in at most $4n/d$ steps, and the number of edges removed is at most $(4n/d) \cdot d/4 = n$. ■

Proof of Theorem 173: The proof follows by putting together lemmas 174 and 175. ■

We overload notation here by denoting $W(B,A) = W(P \setminus A, Q \setminus B) = C(A,B) \setminus W(A,B)$ for $A \subseteq P, B \subseteq Q$. The main result of this subsection is

Corollary 176 *Let $G^* = (P,Q,E^*)$ be a graph obtained by sampling the edges of a d-regular bipartite graph $G = (P,Q,E)$ on $2n$ vertices independently with probability p. There exists a constant $c > 0$ such that if $p > \frac{c \ln n}{\epsilon^2 d}$ then whp for all pairs $(A,B), A \subseteq P, B \subseteq Q$, $|A| \geq |B| + 2n/d$ one has that $|W(A,B) \cap E^*| > \frac{1-3\epsilon}{1+2\epsilon}|W(B,A) \cap E^*|$ for all $\epsilon < 1/4$. In particular, G^* contains a matching of size at least $n - 2n/d$ whp.*

Proof: Set $A_i = A \cap P_i, B_i = B \cap Q_i$, where $G_i = (P_i, Q_i, E_i)$ are the pieces of the decomposition obtained in Section 7.4.2. For each (A_i, B_i) such that G_i is not an isolated vertex we have by Lemma 174 and Theorem 172

$$||W_{G_i}(A_i,B_i) \cap E^*| - p|W_{G_i}(A_i,B_i)|| < \epsilon p |C_{G_i}(A_i,B_i)|.$$

If G_i is an isolated vertex, we have $|W_{G_i}(A_i,B_i) \cap E^*| = p|W_{G_i}(A_i,B_i)| = 0$. Since the latter estimate is stronger than the former, we shall not consider the isolated vertices separately in what follows.

Adding these inequalities over all i we get

$$\sum_{i=1}^{k} |W_{G_i}(A_i,B_i) \cap E^*| \geq p \sum_{i=1}^{k} |W_{G_i}(A_i,B_i)| - \epsilon p \sum_{i=1}^{k} |C_{G_i}(A_i,B_i)|. \tag{7.8}$$

Denote the set of edges removed during the decomposition process by E_r. Denote $E_1 = E_r \cap W(A,B)$ and $E_2 = E_r \cap W(B,A)$. Since $|W(A,B) \cap E^*| = \sum_{i=1}^{k} |W_{G_i}(A_i,B_i) \cap E^*| + |E_1 \cap E'|$ and $\sum_{i=1}^{k} |W_{G_i}(A_i,B_i)| = |W(A,B)| - |E_1|$, this implies

$$|W(A,B) \cap E^*| \geq p|W(A,B)| - \epsilon p |C(A,B)| - p|E_1|.$$

Likewise, since $W(B,A) = W(P \setminus A, Q \setminus B)$, we have

$$|W(B,A) \cap E^*| \leq p|W(B,A)| + \epsilon p |C(A,B)| + p|E_2|.$$

Since $|A| \geq |B| + 2n/d$, we have $|W(A,B)| \geq |W(B,A)| + 2n$, so

$$\begin{aligned}
|W(A,B) \cap E^*| &\geq p|W(A,B)| - \epsilon p |C(A,B)| - p|E_1| \\
&\geq p(|W(B,A)| + 2n) - \epsilon p |C(A,B)| - p|E_1| - p|E_2| \\
&\geq |W(B,A) \cap E^*| - 2\epsilon p |C(A,B)| + p(2n - |E_r|) \\
&\geq |W(B,A) \cap E^*| - 2\epsilon p |C(A,B)| + pn.
\end{aligned}$$

By similar arguments $|C(A,B) \cap E^*| \geq (1-\epsilon)p(|C(A,B)| - n)$, i.e. $p|C(A,B)| \leq \frac{1}{1-\epsilon}|C(A,B) \cap E^*| + pn$. Hence, we have

$$\begin{aligned}|W(A,B) \cap E^*| &\geq |W(B,A) \cap E^*| - 2\epsilon p|C(A,B)| + pn \\ &\geq |W(B,A) \cap E^*| - \frac{2\epsilon}{1-\epsilon}|C(A,B) \cap E^*| + (1-2\epsilon)pn \\ &\geq |W(B,A) \cap E^*| - \frac{2\epsilon}{1-\epsilon}\left(|W(A,B) \cap E^*| + |W(B,A) \cap E^*|\right) + (1-2\epsilon)pn,\end{aligned}$$

which implies

$$|W(A,B) \cap E^*| > \frac{1-3\epsilon}{1+\epsilon}|W(B,A) \cap E^*|$$

for $\epsilon < 1/4$. This completes the proof. ■

Remark 177 *The result in corollary 176 is tight up to an $O(\ln d)$ factor for $d = \Omega(\sqrt{n})$.*

Proof: The following construction gives a lower bound of $n - \Omega\left(\frac{n}{d \ln d}\right)$. Denote by $G_{n,d}$ the graph from Theorem 8 in Chapter 2 and denote by $G^*_{n,d}$ a graph obtained by sampling edges of $G_{n,d}$ at the rate of $\frac{c \ln n}{d}$ for a constant $c > 0$. Define the graph G as d disjoint copies of $G_{2d \ln d,d}$, and denote the sampled graph by G^*. Note that by Theorem 4.1 the maximum matching in each copy of $G^*_{2d \ln d,d}$ has size at most $2d \ln d - 1$ whp, and since the number of vertices in G is $N = 2d^2 \ln d$, the maximum matching in G^* has size at most $N - \Omega\left(\frac{N}{d \ln d}\right)$ whp. ■

7.4.3 Runtime analysis of the Hopcroft-Karp algorithm

In this section we derive a bound on the runtime of the Hopcroft-Karp algorithm on the subsampled graph obtained in step **S2** of our algorithm. The main object of our analysis is the alternating level graph, which we now define. Given a partial matching of a graph $G = (P, Q, E)$, the alternating level graph is defined inductively. Define sets A_j and B_j, $j = 1, \ldots, L$ as follows. Let A_0 be the set of unmatched vertices in P and let $B_0 = \emptyset$. Then let $B_{j+1} = \Gamma(A_j) \setminus \left(\bigcup_{i<j} B_i\right)$, where $\Gamma(A)$ is the set of neighbours of vertices in $A \subseteq V(G)$, and let A_j be the set of vertices matched to vertices from B_j. The construction terminates when either B_{j+1} contains an unmatched vertex or when $B_{j+1} = \emptyset$, and then we set $L = j$. We use the notation $A^{(j)} = \bigcup_{k \leq j} A_k, B^{(j)} = \bigcup_{k \leq j} B_k$. We now give an outline of the Hopcroft-Karp algorithm for convenience of the reader. Given a non-maximum matching, the algorithm starts by constructing the alternating level graph described above and stops when an unmatched vertex is found. Then the algorithm finds a maximal set of vertex-disjoint augmenting paths of length L (this can be done by depth-first search in $O(m)$ time) and performs the augmentations, thus completing one augmentation phase. It can be shown that each augmentation phase increases the length of the shortest augmenting path. Standard analysis of the run-time for general bipartite graphs is based on the observation that once $\sqrt{n}$ augmentations have been performed, the constructed matching necessarily has size at most $\sqrt{n}$ smaller than the maximum matching.

We denote the graph obtained by sampling edges of G independently with probability $p = \frac{c \ln n}{d}$ for a constant $c > 0$ by G^*. Note that G^* is obtained from G by *uniform* sampling. We will make the connection to non-uniform sampling in Theorem 180. For $A \subseteq V(G)$ denote the set of edges in the cut $(A, V(G) \setminus A)$ in G by $\delta(A)$ and the set of edges in the same cut in G^* by $\delta^*(A)$. Similarly, we denote the vertex neighbourhood of A in G by $\Gamma(A)$ and the vertex neighbourhood in G^* by $\Gamma^*(A)$. We consider the alternating level graph in G^* and prove that whp for any partial matching of size smaller than $n - 2n/d$ for each $1 \leq j \leq L$ either $|B_{j-1} \cup B_j \cup B_{j+1}| = \Omega(d)$ or B_j expands by at least a factor of $\ln n$ in either forward or backward direction ($|B_{j+1}| \geq (\ln n)|B_j|$ or $|B_{j-1}| \geq (\ln n)|B_j|$). This implies that $L = O\left(\frac{n \ln d}{d \ln \ln n}\right)$, thus yielding the same bound on the length of the shortest augmenting path by virtue of corollary 176. The main technical result of this subsection is

Lemma 178 *Let the graph G^* be obtained from the bipartite d-regular graph G on $2n$ vertices by uniform sampling with probability p. There exist constants $c > 0, \epsilon > 0$ such that if $p \geq \frac{c \ln n}{\epsilon^2 d}$, then whp for any partial matching in G^* of size smaller than $n - 2n/d$ there exists an augmenting path of length $O\left(\frac{n \ln d}{d \ln \ln n}\right)$.*

The following expansion property of the graph G^* will be used to prove lemma 178:

Lemma 179 *Define $\gamma(t) = (1 - \exp(-t))/t$. For all $t > 0$ there exists a constant $c > 0$ that depends on t and ϵ such that if G^* is obtained by sampling the edges of G independently with probability $p > \frac{c \ln n}{d}$, then whp for every set $A \subseteq P$, $|A| \leq t/p$ (resp. $B \subseteq Q$, $|B| \leq t/p$)*

$$|\Gamma^*(A)| \geq (1 - \epsilon) dp\gamma(t)|A|.$$

Proof: Consider a set $A \subseteq P$, $|A| \leq t/p$. For $b \in \Gamma(A)$ denote the indicator variable corresponding to the event that at least one edge incident on b and going to A is sampled by X_b, i.e. $X_b = I_{\{b \in \Gamma^*(A)\}}$. Denote the number of edges between b and vertices of A by k_b. We have

$$\mathbf{Pr}[X_b = 1] = 1 - (1 - p)^{k_b} \geq 1 - \exp(-k_b p) \geq k_b p \gamma(t),$$

since $k_b p \leq t$ and $e^{-x} \leq 1 - \gamma(t)x$ for $x \in [0, t]$.

Hence,

$$\mathbf{E}\left[\sum_{b \in B} X_b\right] \geq p \sum_{b \in B} k_b \geq p|\delta(A)|\gamma(t). \tag{7.9}$$

There are at most n^s subsets A of P of size s and $|\delta(A)| = d|A|$ for all A, so we obtain using Chernoff bounds and the union bound

$$\begin{aligned}\mathbf{Pr}\left[\exists\ A \subseteq P : |\Gamma^*(A)| < (1 - \epsilon)pd|A|\gamma(t)\right] &< \sum_{s=1}^{n} n^s \exp\left(-\epsilon^2 pds\gamma(t)\right) \\ &= \sum_{s=1}^{n} \exp\left(s(1 - c\gamma(t)) \ln n\right) = O(n^{2 - c\gamma(t)}),\end{aligned}$$

which can be made $O(n^{-r})$ by choosing $c > (2+r)/\gamma(t)$ for any $r > 0$. ■

Proof of Lemma 178: First note that since the partial matching is of size strictly less than $n - 2n/d$, by Corollary 176 there exists an augmenting path with respect to the partial matching.

In order to upperbound the length of the shortest augmenting path, we will show that for each j, at least one of the following is true:

1. $|B_j| \geq d/500$;
2. $|B_{j+1}| \geq d/500$;
3. $|B_{j+1}| \geq (\ln n)|B_j|$;
4. $|B_{j-1}| \geq d/500$;
5. $|B_{j-1}| \geq (\ln n)|B_j|$.

It then follows that for each j there exists j' such that $|j - j'| \leq 1 + \ln_{\ln n} d$ and $|B_{j'}| \geq d/500$. Hence, there cannot be more than $O\left(\frac{n \ln d}{d \ln \ln n}\right)$ levels in the alternating level graph, so there always exists an augmenting path of length $O\left(\frac{n \ln d}{d \ln \ln n}\right)$.

For each $1 \leq j \leq L$, where L is the number of levels in the alternating level graph, we classify the edges leaving B_j into three classes: (1) E_F contains edges that go to $U \setminus A^{(j)}$, (2) E_M contains edges that go to A_j, and (3) E_R contains edges that go to A_{j-1}. At least one of E_F, E_M, E_R has at least $(1-\epsilon)pd|B_j|/3$ edges by Lemma 179. We now consider each of these possibilities.

Case (A): First suppose that E_F contains at least $(1-\epsilon)pd|B_j|/3$ edges. Note that since the partial matching has size smaller than $n - 2n/d$ by assumption, we have that $|A^{(j)}| \geq |B^{(j)}| + 2n/d$. Hence, by Corollary 176 the number of edges going from A_j to B_{j+1} is at least

$$\frac{(1-3\epsilon)(1-\epsilon)}{1+\epsilon} pd|A_j|/3.$$

Suppose first that $|A_j| < 1/(5p)$. Then by Lemma 179 one has that $|\Gamma^*(A_j)| \geq (1-\epsilon)\gamma(1/5)pd|A_j|$. Let $\beta^* = 1 + \epsilon - \frac{(1-3\epsilon)(1-\epsilon)}{3(1+\epsilon)}$. Observe that since one edge going out of A_j yields at most one neighbor, at most $(1+\epsilon)pd|A_j| - \frac{(1-3\epsilon)(1-\epsilon)}{1+\epsilon} pd|A_j|/3 = \beta^* pd|A_j|$ neighbours of vertices of A_j are outside B_{j+1}. Setting $\epsilon = 1/15$, we get that B_{j+1} contains at least $((1-\epsilon)\gamma(1/5) - \beta^*)pd|A_j| > 0.011pd|A_j| > (\ln n)|A_j|$ neighbours of $|A_j|$, i.e. $|B_{j+1}| \geq (\ln n)|A_j| = (\ln n)|B_j|$ (this corresponds to case 3 above). Now if $|A_j| \geq 1/(5p)$, one can find $A' \subseteq A_j$ such that $|A'| = \lfloor 1/(5p) \rfloor$ and at least $\frac{(1-3\epsilon)(1-\epsilon)}{1+\epsilon} pd|A'|/3$ edges going out of A' go to B_{j+1}, which implies by the same argument that $|B_{j+1}| \geq 0.011pd|A'| \geq d/500$ (this corresponds to case 2 above).

Case (B): Suppose that E_M contains at least $(1-\epsilon)pd|B_j|/3$ edges. Then by the same argument as in the previous paragraph (after first weakening our estimate to $\frac{(1-3\epsilon)(1-\epsilon)}{1+\epsilon} pd|B_j|/3$) we have that $|A_j| \geq (\ln n)|B_j|$ if $|B_j| \leq 1/(5p)$. This is impossible when $\ln n > 1$ since $|A_j| = |B_j|$. Hence, $|A_j| \geq d/500$ by same argument as above, and hence $|B_j| \geq d/500$ (this corresponds to case 1 above).

Case (C): Suppose that E_R contains at least $(1-\epsilon)pd|B_j|/3$ edges. By the same argument as above we have that either $|B_{j-1}| \geq d/500$ (this corresponds to case 5 above) or $B_{j-1} \geq (\ln n)|B_j|$ (this corresponds to case 4 above).

This completes the proof. ■

We can now prove the main result of this section:

Theorem 180 *Let the graph G^* be obtained from G using steps **S1** and **S2** in the algorithm of section 7.3. Then step **S3** takes $O\left(\frac{n^2 \ln^2 n}{d \ln \ln n}\right)$ time whp, giving a time of $O\left(\frac{n^2 \ln^3 n}{d}\right)$ for the entire algorithm.*

Proof: We analyze the runtime of step **S3** in two stages: (1) finding a matching of size $n - 2n/d$, and (2) extending the matching of size $n - 2n/d$ to a perfect matching.

Note that the strength of edges in G' obtained after **S1** does not exceed $\frac{(1+\epsilon)cn \ln n}{\epsilon^2 d}$, the maximum degree, with high probability, for a constant $c > 0$. Hence, the combination of sampling uniformly in **S1** and non-uniformly in **S2** dominates sampling each edge with probability $\Omega\left(\frac{\ln n}{d}\right)$, so we write $G'' = (P, Q, E^* \cup E^{**})$, where E^* is obtained from E by sampling uniformly with probability $p = \frac{c \ln n}{d}$ for a sufficiently large $c > 0$. The constant $c > 0$ can be made sufficiently large so that Lemma 178 applies by adjusting the constant in the sampling in steps **S1** and **S2**. Denote $G^* = (P, Q, E^*)$ and note that the proof of Lemma 178 only uses lower bounds on the number of edges incident to vertices in a given set, as well as the number of vertex neighbours of a set of vertices. Hence, since all bounds apply to G^*, the conclusion of the lemma is valid for G'' whp as well, and we conclude that the maximum number of layers in an alternating level graph, and hence the length of the shortest augmenting path, is $O\left(\frac{n \ln d}{d \ln \ln n}\right)$. As each augmentation phase takes time proportional to the number of edges in the graph, this implies that the first stage takes $O\left(\frac{n^2 \ln^2 n}{d \ln \ln n}\right)$.

Finally, note that each augmentation phase increases the size of the matching by at least 1, and thus $O(n/d)$ augmentation suffice to extend the matching constructed in the first stage to a perfect matching. This takes $O\left(\frac{n^2 \ln n}{d}\right)$ time, so the runtime is $O\left(\frac{n^2 \ln^2 n}{d \ln \ln n}\right)$ for step **S3**, and $O\left(\frac{n^2 \ln^3 n}{d}\right)$ overall. ■

Remark 181 *Theorem 180 as well as lemma 178 can be slightly altered to show that the runtime of the Hopcroft-Karp algorithm on the uniformly subsampled graph in Chapter 2 is $O\left(\frac{n^3 \ln^2 n}{d^2 \ln \ln n}\right)$. This shows that the uniform sampling approach from Chapter 2 yields an $\tilde{O}(n^{5/3})$ algorithm, which is better than $\tilde{O}(n^{1.75})$ obtained there.*

Theorem 182 *For any function $d(n) \geq 2\sqrt{n}$ there exists an infinite family of $d(n)$-regular graphs with $2n + o(n)$ vertices such that whp the algorithm in section 7.3 performs $\Omega(n/d)$ augmentations in the worst case.*

Proof: In what follows we omit the dependence of d on n for brevity. Define $H^{(k)} = (U, V, E)$, $0 \leq k \leq d$, to be a $(d-k)$-regular bipartite graph with $|U| = |V| = d$. The graph G consists of t copies of $H^{(k)}$, which we denote by $\{H_j\}_{j=1}^t$, where $H_j = H^{(t-j+1)}$, and $2t$ vertices $u_1, \ldots, u_t$ and $v_1, \ldots, v_t$. Each of $u_1, \ldots, u_t$ is connected to all d vertices in the V-part of H_1, and for $1 \leq j \leq t$, the

vertex v_j is connected to all vertices in the U-part of H_j. The remaining connections are established by adding $t-j$ edge-disjoint perfect matchings between the U part of H_j and the V part of H_{j+1} for all $1 \leq j < t$.

Set $t = n/d \leq \sqrt{n}/2 \leq d/4$. Note that the strength of edges in H_j is at least $d/4$, so whp there exists a perfect matching in subgraph of H_j generated by the sampling steps **S1** and **S2**, for $1 \leq j \leq t$. Suppose that at the first iteration of the Hopcroft-Karp algorithm a perfect matching is found in each H_j, thus leaving unmatched the vertices $u_1, \ldots, u_t$ and $v_1, \ldots, v_t$. Then from this point on, the shortest augmenting path for each pair pair (u_j, v_j) has length j, and each augmentation phase of the Hopcroft-Karp algorithm will increase the size of the matching by 1. Hence, it takes t augmentations to find a perfect matching. The number of vertices is $2(d+1)t = 2n + o(n)$. ■

7.5 Perfect Matchings in Doubly Stochastic Matrices

An $n \times n$ matrix A is said to be *doubly stochastic* if every element is non-negative, and every row-sum and every column-sum is 1. The celebrated Birkhoff-von Neumann theorem says that every doubly stochastic matrix is a convex combination of permutation matrices (*i.e.*, matchings). Surprisingly, the running time of computing this convex combination (known as a Birkhoff-von Neumann decomposition) is typically reported as $O(m^2\sqrt{n})$, even though much better algorithms can be easily obtained using existing techniques or very simple modifications. We list these running times here since there does not seem to be any published record[5]. After listing the running times that can be obtained using existing techniques, we will show how proportionate uncrossings can be applied to this problem to obtain a slight improvement.

1. An $O(m^2)$-time algorithm for finding a Birkhoff-von Neumann decomposition can be obtained by finding a perfect matching in the existing graph using augmenting paths (in time $O(mn)$), assigning this matching a weight which is the weight of the smallest edge in the matching, subtracting this weight from every edge in the matching (causing one or more edges to be removed from the support of A), and continuing the augmenting path algorithm without restarting. When a matching is found, if we remove k edges then we need to find only k augmenting paths (finding each augmenting path takes time $O(m)$) to find another matching, which leads to a total time of $O(m^2)$.

2. Let b be the maximum number of significant bits in any entry of A. An $O(mb)$-time algorithm for finding a single perfect matching in the support of a doubly stochastic matrix can be easily obtained using the technique of Gabow and Kariv [33]: repeatedly find Euler tours in edges where the lowest order bit (say bit j) is 1, and then increase the weight of all edges going from left to right by 2^{-j} and decrease the weight of all edges going from right to left by the same amount, where the directionality of edges corresponds to an arbitrary orientation of the

[5]This list was compiled by Bhattacharjee and Goel and is presented here to provide some context rather than as original work.

Euler tour; this eliminates bit j while preserving the doubly stochastic property and without increasing the support.

3. An $O(mnb)$-time algorithm to compute the Birkhoff-von Neumann decomposition can be obtained using the edge coloring algorithm of Gabow and Kariv [33].

We now show how our techniques lead to an $O(m \ln^3 n + n^{1.5} \ln n)$-time algorithm for finding a single perfect matching in the support of a doubly stochastic matrix. In realistic scenarios, this is unlikely to be better than (2) above, and we present this primarily to illustrate another application of our proportionate uncrossing technique. First, define a weighted bipartite graph $G = (P, Q, E)$, where $P = \{u_1, u_2, \ldots u_n\}$ corresponds to rows of A, $Q = \{v_1, v_2, \ldots, v_n\}$ corresponds to columns of A, and $(u_i, v_j) \in E$ iff $A_{i,j} > 0$. Define a weight function w on edges, with $w(u_i, v_j) = A_{i,j}$. Let $\mathcal{R}$ be the collection of all pairs $(A, B), A \subseteq P, B \subseteq Q, |P| > |Q|$. Since A is doubly stochastic, the collection $\mathcal{R}$ is (1/2)-thick with respect to (G, w, E). Let $\mathcal{T}$ be a (1/2)-uncrossing of $\mathcal{R}$. Performing a Benczúr-Karger sampling on G will guarantee (with high probability) that at least one edge is sampled from every witness set in $W(\mathcal{T})$, and hence running the Hopcroft-Karp algorithm on the sampled graph will yield a perfect matching with high probability. The running time of $O(m \ln^3 n + n^{1.5} \ln n)$ is just the sum of the running times of Benczúr-Karger sampling for weighted graphs [16] and the Hopcroft-Karp matching algorithm [45].

7.6 Proof of Lemma 154

Consider any (A, B) where $|A| > |B|, A \subseteq P, B \subseteq Q$. Define $A_i = P_i \cap A$ and $B_i = Q_i \cap B$. Fix an i such that $|A_i| > |B_i|$; such an i is guaranteed to exist. By the definition of relevance, there exists a pair $(X, Y) \in \mathcal{R}$ such that $X \subseteq A_i$, and $W(X, Y) \cap E_R \subseteq W(A_i, B_i) \cap E_R$. By the assumption in the theorem, there exists an edge $(u, v) \in E^* \cap E_R \cap W(X, Y)$. Since $W(X, Y) \cap E_R \subseteq W(A_i, B_i) \cap E_R$, it follows that $(u, v) \in E^* \cap E_R \cap W(A_i, B_i)$. This edge is in G^*, and goes from A_i to $Q_i \setminus B_i$, *i.e.*, from A_i to $Q_i \setminus (Q_i \cap B)$, and hence, from A to $Q \setminus B$. Since the only assumption on (A, B) was that $|A| > |B|$, we can now invoke Hall's theorem to claim that G^* has a perfect matching. ■

7.7 Proof of Theorem 157

As mentioned before, the proof is along very similar lines to that of the Benczúr-Karger sampling theorem, but does not follow in a black-box fashion and is presented here for completeness. The proof relies on the following result due to Karger and Stein [55]:

Lemma 183 *Let $H(V, E)$ be an undirected graph on n vertices such that each edge e has an associated non-negative weight $\tilde{p}_e$. Let s^* be the value of minimum cut in H under the weight function $\tilde{p}_e$. Then for any $\alpha \geq 1$, the number of cuts in H of weight at most αs^* is less than $n^{2\alpha}$.*

Proof of Theorem 157: We will choose $c = 5$. The first part of the proof shows that it is sufficient to bound a certain expression that involves only cuts. The second part then bounds this expression.

For the first part, let $\mu(X) = \sum_{e \in X} p_e$ denote the expected number of edges chosen from X by the sampling process. If a set $X \in \mathcal{X}$ contains an edge e with $p_e = 1$, then that edge will definitely be chosen, and that set does not contribute to

$$\sum_{X \in \mathcal{X}} \Pr[\text{No edge in } X \text{ is chosen in } H']$$

and can be removed from $\mathcal{X}$. Hence, assume without loss of generality that $p_e < 1$ for every edge in $\bigcup_{X \in \mathcal{X}} X$. Define $\tilde{\mu}(X) = \sum_{e \in X} \left(\frac{c \ln n}{s_e} \right)$. Now for any set $X \in \mathcal{X}$,

$$\Pr[\text{No edge in } X \text{ is chosen in } H'] = \prod_{e \in X} (1 - p_e) \leq \prod_{e \in X} e^{-p_e} \leq e^{-\mu(X)},$$

where

$$\mu(X) = \frac{c \ln n}{\gamma} \sum_{e \in X} \frac{1}{s_e} > (c \ln n) \sum_{e \in f(X)} \frac{1}{s_e} = \tilde{\mu}(f(X)).$$

Since f is a one-one function, it is sufficient to provide an upper-bound on $\sum_{C \in \mathcal{C}} e^{-\tilde{\mu}(C)}$.

For the second part, let $\tilde{\mu}_1, \tilde{\mu}_2, \ldots, \tilde{\mu}_{2^n - 2}$ be a non-decreasing sorted sequence corresponding to the multi-set $\{\tilde{\mu}(C) : C \in \mathcal{C}\}$. Define $q_i = e^{-\tilde{\mu}_i}$. Consider an arbitrary cut C. Any edge in C can have strength at most $|C|$, and hence $\tilde{\mu}(C) \geq c \ln n$, and therefore, $q_1 \leq n^{-c}$. So the sum of q_i for the first n^2 cuts in the sequence is bounded by n^{-c+2}. We now focus on the remaining cuts. By Lemma 183, we know that for any $\alpha \geq 1$, we have $\tilde{\mu}_{n^{2\alpha}} \geq \alpha \tilde{\mu}_1$. Hence

$$\tilde{\mu}_k \geq \frac{\ln k}{2 \ln n} \tilde{\mu}_1,$$

which in turn implies that $q_k \leq k^{-c/2}$. Thus

$$\sum_{X \in \mathcal{X}} \Pr[\text{No edge in } X \text{ is chosen in } H'] \leq \sum_{C \in \mathcal{C}} e^{-\tilde{\mu}(C)} \leq \sum_{k=1}^{n^2} q_k + \sum_{k > n^2} q_k \leq n^{-c+2} + \sum_{k > n^2} k^{-c/2} = O(n^{-c+2}),$$

giving us the desired result when we choose $c = 5$. ∎

7.8 Proof of Lemma 159

Assume by way of contradiction that no such integer j exists for some pair of multisets S_1 and S_2. Let K be the largest integer in $S_1 \cup S_2$, and let α_i and β_i denote the number of occurrences of i in the multisets S_1 and S_2 respectively. Then for all $j \geq 1$, we have

$$\sum_{i=j}^{K} \frac{\alpha_i}{i} \leq \gamma \left(\sum_{i=j}^{K} \frac{\beta_i}{i} \right).$$

Summing the above inequality for all $j \in \{1..K\}$, we get

$$\sum_{i=1}^{K} \alpha_i \leq \gamma \left(\sum_{i=1}^{K} \beta_i \right),$$

which is a contradiction since $|S_1| > \gamma |S_2|$ by assumption. ■

Bibliography

[1] Gagan Aggarwal, Rajeev Motwani, Devavrat Shah, and An Zhu. Switch scheduling via randomized edge coloring. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, 2003.

[2] K. Ahn and S. Guha. Graph sparsification in the semi-streaming model. *Automata, Languages and Programming*, pages 328–338, 2009.

[3] K. Ahn and S. Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *ICALP*, pages 526–538, 2011.

[4] K. Ahn and S. Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *CoRR*, abs/1104.2315, 2011.

[5] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. *SODA*, pages 459–467, 2012.

[6] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketching: Sparsification, spanners, and subgraphs. *To appear in proceedings of 2012 ACM PODS*, 2012.

[7] Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993.

[8] Surender Baswana. Streaming algorithm for graph spanners - single pass and constant processing time per edge. *Inf. Process. Lett.*, 106(3):110–114, 2008.

[9] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (alpha, beta)-spanners. *ACM Transactions on Algorithms*, 7(1), 2010.

[10] Surender Baswana and Sandeep Sen. A simple linear time algorithm for computing a (2k-1)-spanner of o($n^{1+1/k}$) size in weighted graphs. *ICALP*, pages 384–296, 2003.

[11] Surender Baswana and Sandeep Sen. Approximate distance oracles for unweighted graphs in expected $o(n^2)$ time. *ACM Transactions on Algorithms*, 2(4):557–577, 2006.

[12] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *STOC*, pages 255–262, 2009.

[13] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *STOC*, pages 255–262, 2009.

[14] F. A. Behrend. On sets of integers which contain no three terms in arithmetic progression. *Proc. Nat. Acad. Sci.*, 32:331–332, 1946.

[15] A. Benczur. Cut structures and randomized algorithms in edge-connectivity problems. *PhD Thesis*, 1997.

[16] András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. *Proceedings of the 28th annual ACM symposium on Theory of computing*, pages 47–55, 1996.

[17] G. Birkhoff. Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucumán Rev. Ser. A*, 5:147–151, 1946.

[18] B. Bollobas. *Modern graph theory*. Springer, 1998.

[19] E.J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, 2006.

[20] C. Chang, D. Lee, and Y. Jou. Load balanced birkhoff-von neumann switches, part i: one-stage buffering. *Computer Communications*, 25(6):611–622, 2002.

[21] Denis Xavier Charles, Max Chickering, Nikhil R. Devanur, Kamal Jain, and Manan Sanghi. Fast algorithms for finding matchings in lopsided bipartite graphs with applications to display ads. *ACM Conference on Electronic Commerce*, pages 121–128, 2010.

[22] R. Cole and J.E. Hopcroft. On edge coloring bipartite graphs. *SIAM J. Comput.*, 11(3):540–546, 1982.

[23] R. Cole, K. Ost, and S. Schirra. Edge-coloring bipartite multigraphs in $O(E \log D)$ time. *Combinatorica*, 21(1):5–12, 2001.

[24] G. Cormode and S. Muthukrishnan. Combinatorial algorithms for compressed sensing. *Structural Information and Communication Complexity*, 4056:280–294, 2006.

[25] D.L. Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[26] 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*, volume 5757 of *Lecture Notes in Computer Science*, pages 492–503. Springer Berlin / Heidelberg, 2009.

[27] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348:207–216, 2005.

[28] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348:207–216, 2005.

[29] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. *STOC*, 2002.

[30] Wai Shing Fung, Ramesh Hariharan, Nicholas J. A. Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. *STOC*, pages 71–80, 2011.

[31] Wai Shing Fung, Ramesh Hariharan, Nicholas J. A. Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. *STOC*, pages 71–80, 2011.

[32] Harold N. Gabow. Using euler partitions to edge color bipartite multigraphs. *International Journal of Parallel Programming*, 5(4):345–355, 1976.

[33] H.N. Gabow and O. Kariv. Algorithms for edge coloring bipartite graphs and multigraphs. *SIAM J. Comput.*, 11(1):117–129, 1982.

[34] A. Gilbert and P. Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010.

[35] A. Goel, M. Kapralov, and S. Khanna. On the communication and streaming complexity of maximum bipartite matching. *(preliminary version appeared in SODA 2012)*.

[36] A. Goel, M. Kapralov, and S. Khanna. Perfect matchings in $\tilde{O}(n^{1.5})$ time in regular bipartite graphs. *http://arxiv.org/abs/0902.1617v2*, 2009.

[37] A. Goel, A. Meyerson, and S. Plotkin. Approximate majorization and fair online load balancing. *ACM Transactions on Algorithms*, 1(2):338–349, Oct 2005.

[38] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings via uniform sampling in regular bipartite graphs. *ACM Trans. Algorithms (Preliminary version appeared in SODA'09)*, 6(2):27:1–27:13, April 2010.

[39] Ashish Goel, Michael Kapralov, and Ian Post. Single pass sparsification in the streaming model with edge deletions. *http://arxiv.org/abs/1203.4900*, 2012.

[40] T. W. Gowers. Some unsolved problems in additive/combinatorial number theory. *http://www.dpmms.cam.ac.uk/ wtg10/addnoth.survey.dvi.*

[41] Navin Goyal, Luis Rademacher, and Santosh Vempala. Expanders via random spanning trees. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 576–585, 2009.

[42] Torben Hagerup, Kurt Mehlhorn, and J. Ian Munro. Maintaining discrete probability distributions optimally. *ICALP*, pages 253–264, 1993.

[43] J. Hastad and A. Wigderson. Simple analysis of graph tests for linearity and pcp. *Random Structures and Algorithms*, 22, 2003.

[44] M.R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. In *External memory algorithms: DIMACS Workshop External Memory and Visualization, May 20-22, 1998*, volume 50, pages 107–118. American Mathematical Society, 1999.

[45] J.E. Hopcroft and R.M. Karp. An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[46] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Proceedings of 41st Annual Symposium on Foundations of Computer Science*, pages 189–197. IEEE, 2000.

[47] Ajai Kapoor and Romeo Rizzi. Edge-coloring bipartite graphs. *J. Algorithms*, 34(2):390–396, 2000.

[48] Michael Kapralov. Better bounds for matchings in the streaming model. *http://arxiv.org/abs/1206.2269*, 2012.

[49] Michael Kapralov and Rina Panigrahy. Spectral sparsification via random spanners. *ITCS*, pages 393–398, 2012.

[50] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. *STOC*, pages 587–596, 2011.

[51] D. Karger. Using randomized sparsification to approximate minimum cuts. *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 424–432, 1994.

[52] D. Karger. Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research (Preliminary version appeared in the Proceedings of the 26th annual ACM symposium on Theory of computing)*, 24(2):383–413, 1999.

[53] D. Karger and M. Levine. Random sampling in residual graphs. *STOC*, 2002.

[54] D. Karger and M. Levine. Random sampling in residual graphs. *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 2002.

[55] David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *J. ACM*, 43(4):601–640, 1996.

[56] R. Karp, U. Vazirani, and V. Vazirani. An optimal algorithm for online bipartite matching. *STOC*, 1990.

[57] Jonathan A. Kelner and Alex Levin. Spectral sparsification in the semi-streaming setting. *STACS*, pages 440–451, 2011.

[58] R. Khandekar, S. Rao, and U. Vazirani. Graph partitioning using single commodity flows. *Journal of the ACM (JACM)*, 56(4):19:1–19:15, 2009.

[59] J. Kleinberg, Y. Rabani, and E. Tardos. Fairness in routing and load balancing. *J. Comput. Syst. Sci.*, 63(1):2–20, 2001.

[60] Alexandra Kolla, Yury Makarychev, Amin Saberi, and Shang-Hua Teng. Subgraph sparsification and nearly optimal ultrasparsifiers. *STOC*, pages 57–66, 2010.

[61] D. König. Uber graphen und ihre anwendung auf determinententheorie und mengenlehre. *Math. Annalen*, 77:453–465, 1916.

[62] Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. *CoRR*, abs/1112.0184, 2011.

[63] Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving sdd linear systems. In *FOCS*, pages 235–244, 2010.

[64] Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving sdd linear systems. *FOCS*, pages 235–244, 2010.

[65] Ioannis Koutis, Gary L. Miller, and Richard Peng. A nearly-m log n time solver for sdd linear systems. In *FOCS*, pages 590–598, 2011.

[66] V. I. Levenstein. Upper bounds for codes with a fixed weight of vectors (in russian). *Problems of information transmission*, pages 3–12, 1971.

[67] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. *STOC*, pages 597–606, 2011.

[68] A. McGregor. Finding graph matchings in data streams. *APPROX-RANDOM*, pages 170–181, 2005.

[69] A. McGregor. Graph mining on streams. *Encyclopedia of Database Systems*, pages 1271–1275, 2009.

[70] Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. *FOCS*, pages 109–118, 2006.

[71] R. Motwani. Average-case analysis of algorithms for matchings and related problems. *Journal of the ACM(JACM)*, 41:1329–1356, 1994.

[72] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[73] Rajeev Motwani, Rina Panigrahy, and Ying Xu. Fractional matching via balls-and-bins. *APPROX-RANDOM*, pages 487–498, 2006.

[74] M. Mucha and P. Sankowski. Maximum matchings via gaussian elimination. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 248 – 255, Oct. 2004.

[75] S. Muthukrishnan. *Data streams: algorithms and applications*. Now publishers, 2006.

[76] Marc Najork, Rina Panigrahy, and Yinglian Xie. How user behavior is related to social affinity. *WSDM*, 2012.

[77] S. Raskhodnikova. Property testing: Theory and applications. *Ph.D. thesis*, 2003.

[78] Romeo Rizzi. Finding 1-factors in bipartite regular graphs and edge-coloring bipartite graphs. *SIAM J. Discrete Math.*, 15(3):283–288, 2002.

[79] Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8(2):223–250, 1995.

[80] A. Schrijver. Bipartite edge coloring in $O(\Delta m)$ time. *SIAM J. on Comput.*, 28:841–846, 1999.

[81] A. Schrijver. *Combinatorial Optimization*. Springer Verlag, 2003.

[82] D. Spielman and S. Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

[83] D.A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *STOC*, pages 563–568, 2008.

[84] D.A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. *STOC*, pages 81–90, 2004.

[85] D.A. Spielman and S.H. Teng. Spectral sparsification of graphs. *Arxiv preprint*, 2008.

[86] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, STOC '04, pages 81–90, 2004.

[87] T. Tao and V. Vu. *Additive Combinatorics*. Cambridge University Press, 2009.

[88] H. M. Taylor and S. Karlin. *An Introduction to Stochastic Modeling*. Academic Press, 1998.

[89] M. Thorup and U. Zwick. Approximate distance oracles. *STOC*, 2001.

[90] Mikkel Thorup and Uri Zwick. Spanners and emulators with sublinear distance errors. *SODA*, pages 802–809, 2006.

[91] J. von Neumann. A certain zero-sum two-person game equivalent to the optimal assignment problem. *Contributions to the optimal assignment problem to the Theory of Games*, 2:5–12, 1953.

[92] D. W. Walkup. Matchings in random regular bipartite graphs. *Discrete Math*, 31:59–64, 1980.