New Deterministic Approximation Algorithms for Fully Dynamic Matching*

Sayan Bhattacharya† Monika Henzinger‡ Danupon Nanongkai§

April 19, 2016

Abstract

We present two deterministic dynamic algorithms for the maximum matching problem. (1) An algorithm that maintains a $(2+\varepsilon)$-approximate maximum matching in general graphs with $O(\text{poly}(\log n, 1/\varepsilon))$ update time. (2) An algorithm that maintains an α_K approximation of the *value* of the maximum matching with $O(n^{2/K})$ update time in bipartite graphs, for every sufficiently large constant positive integer K. Here, $1 \leq \alpha_K < 2$ is a constant determined by the value of K. Result (1) is the first deterministic algorithm that can maintain an $o(\log n)$-approximate maximum matching with polylogarithmic update time, improving the seminal result of Onak et al. [STOC 2010]. Its approximation guarantee almost matches the guarantee of the best *randomized* polylogarithmic update time algorithm [Baswana et al. FOCS 2011]. Result (2) achieves a better-than-two approximation with *arbitrarily small polynomial* update time on bipartite graphs. Previously the best update time for this problem was $O(m^{1/4})$ [Bernstein et al. ICALP 2015], where m is the current number of edges in the graph.

*A preliminary version of this paper will appear in STOC 2016.

†The Institute of Mathematical Sciences, Chennai, India. Email: bsayan@imsc.res.in

‡Faculty of Computer Science, University of Vienna, Austria. Email: monika.henzinger@univie.ac.at. This work was done in part while the author was visiting the Simons Institute for the Theory of Computing. The research leading to this work has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 317532 and from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement number 340506.

§KTH Royal Institute of Technology, Sweden. Email: danupon@gmail.com. Supported by Swedish Research Council grant 2015-04659.

Contents

I EXTENDED ABSTRACT

1 Introduction **1**
1.1 Notations and preliminaries . . . 3

2 General Graphs **3**
2.1 Maintaining a large fractional matching . . . 3
2.2 An overview of our approach . . . 4
2.2.1 An ideal skeleton . . . 4
2.2.2 A degree-splitting procedure . . . 5
2.3 From ideal to approximate skeleton . . . 6
2.4 Maintaing an approximate skeleton: Proof of Theorem 2.9 . . . 7
2.4.1 Handling the insertion/deletion of an edge . . . 8
2.4.2 The subroutine TERMINATE-PHASE(.) . . . 8
2.4.3 Correctness. . . . 8
2.4.4 Analyzing the amortized update time. . . . 8
2.5 Approximation guarantee from approximate skeletons: Proof of Theorem 2.7 . . . 9
2.5.1 Proof of Claim 2.15 . . . 12
2.5.2 Proof of Claim 2.16 . . . 13
2.5.3 Proof of Claim 2.17 . . . 14

3 Bipartite graphs **17**
3.1 $(2+\varepsilon)$-approximation in $O(\sqrt{n}/\varepsilon^2)$ update time . . . 17
3.1.1 Proof of Lemma 3.3 . . . 18
3.2 Better than 2-approximation . . . 19
3.2.1 The main framework: Residual edges . . . 19
3.2.2 Proof of Lemma 3.11 . . . 20
3.3 Extensions . . . 23

4 Open Problems **23**

References **24**

II DYNAMIC ALGORITHM FOR GENERAL GRAPHS: FULL DETAILS **26**

5 Preliminaries **27**
5.1 Setting some parameter values . . . 28
5.2 Skeleton of a graph . . . 29

6 Deriving the approximation guarantee: Proof of Theorem 5.8 **30**
6.1 Some basic notations . . . 33
6.2 Proof of Lemma 6.2 . . . 33
6.3 Proof of Lemma 6.3 . . . 34
6.4 Proof of Lemma 6.4 . . . 35
6.5 Proof of Lemma 6.5 . . . 36

6.6 Proof of Lemma 6.6 . . . 36
6.7 Proof of Lemma 6.8 . . . 37
6.7.1 Outline of the proof . . . 38
6.7.2 The complete proof . . . 38

7 Maintaining the edge-set of a skeleton: Proof of Theorem 5.7 43
7.1 A high level overview of our approach . . . 44
7.2 Critical and laminar structures . . . 44
7.3 Two basic subroutines . . . 47
7.3.1 Proof of Lemma 7.13 . . . 49
7.4 Our algorithm for maintaining critical and laminar structures . . . 51
7.4.1 Initial conditions in the beginning of a phase . . . 52
7.4.2 Handling edge insertion/deletions in the middle of a phase . . . 52
7.4.3 Terminating a phase . . . 54
7.4.4 Some useful properties of our algorithm . . . 55
7.4.5 Proof of Lemma 7.23 . . . 56
7.5 Maintaining the edge-set of an skeleton . . . 57
7.6 Bounding the amortized update time of our algorithm . . . 60
7.6.1 A few notations and terminologies . . . 60
7.6.2 Roadmap . . . 60
7.6.3 A simple bound . . . 60
7.6.4 Analyzing the running time of a single call to REBUILD(j) . . . 61
7.6.5 Bounding the amortized update time of the subroutine REVAMP() . . . 61
7.6.6 Bounding the amortized update time of REBUILD(j) in the middle of a phase . . . 65
7.6.7 Proof of Lemma 7.38 . . . 67

III DYNAMIC ALGORITHM FOR BIPARTITE GRAPHS: FULL DETAILS 70

8 Notations and Preliminaries 71
8.1 An important technical theorem . . . 72

9 Invariants maintained by our algorithm 74
9.1 An overview of the structures maintained by our algorithm . . . 75
9.2 Invariants for levels $i \in \{2,\ldots,K\}$ . . . 77
9.3 Feasibility of the structures for levels $\{2,\ldots,K\}$ . . . 79
9.3.1 Proof of Lemma 9.15 . . . 80
9.3.2 Proof of Lemma 9.17 . . . 82
9.4 Invariants for level $i = 1$ . . . 83
9.5 Feasibility of the structures for level one . . . 83
9.5.1 Proof of Lemma 9.22 . . . 83
9.6 Some useful properties . . . 85
9.7 Feasibility of the solution . . . 86
9.7.1 Proof of Lemma 9.29 . . . 87
9.7.2 Proof of Lemma 9.30 . . . 88
9.7.3 Proof of Lemma 9.31 . . . 90
9.7.4 Proof of Theorem 9.4 . . . 91
9.8 Approximation Guarantee of the Solution . . . 92

9.8.1 Proof of Theorem 9.32 . . . 93
9.8.2 Proof of Lemma 9.35 . . . 97
9.8.3 Proof of Lemma 9.36 . . . 99
9.8.4 Proof of Lemma 9.37 . . . 100

10 The algorithm **102**
10.1 Data structures . . . 103
10.2 Handling the insertion/deletion of an edge (u,v) in the input graph $G=(V,E)$ . . . 104
10.2.1 The subroutine FIX-STRUCTURES(i), where $i \in \{2,\ldots,K\}$. . . . 104
10.3 Analyzing the amortized update time . . . 108
10.3.1 Rules governing the bank accounts . . . 109
10.3.2 Proof of Lemma 10.21 . . . 111
10.4 Maintaining the size of the fractional assignment w_1^*. . . . 113
10.5 Maintaining a $(1+\varepsilon)$-approximation to the size of w^r . . . 114

Part I

EXTENDED ABSTRACT

1 Introduction

In this paper, we consider the *dynamic maximum cardinality matching* problem. In this problem an algorithm has to quickly maintain an (integral) maximum-cardinality matching or its approximation, when the n-node input graph is undergoing edge insertions and deletions. We consider two versions of this problem: In the *matching version*, the algorithm has to output the change in the (approximate) matching, if any, after each edge insertion and deletion. In the *value version*, the algorithm only has to output the value of the matching. (Note that an algorithm for the matching version can be used to solve the value version within the same time.) When stating the running time below, we give the time *per update*[1]. If not stated otherwise, these results hold for both versions.

The state of the art for maintaining an *exact* solution for the value version of this problem is a randomized $O(n^{1.495})$-time algorithm [16]. This is complemented by various hardness results which rules out polylogarithmic update time [1, 8, 11]. As it is desirable for dynamic algorithms to have polylogarithmic update time, the recent work has focused on achieving this goal by allowing *approximate solutions*. The first paper that achieved this is by Onak and Rubinfeld [13], which gave a randomized $O(1)$-approximation $O(\log^2 n)$-time algorithm and a deterministic $O(\log n)$ approximation $O(\log^2 n)$-time algorithm. As stated in the two open problems in [13], this seminal paper opened up the doors for two research directions:

1. Designing a (possibly randomized) *polylogarithmic time* algorithm with smallest approximation ratio.

2. Designing a *deterministic polylogarithmic time* algorithm with constant approximation ratio.

The second question is motivated by the fact that *randomized* dynamic approximation algorithms only fulfill their approximation guarantee when used by an *oblivious* adversary, i.e., an adversary that gives the next update *without* knowing the outputs of the algorithm resulting from earlier updates. This limits the usefulness of randomized dynamic algorithms. In contrast, *deterministic* dynamic algorithms fulfill their approximation guarantee against *any* adversary, even non-oblivous ones. Thus, they can be used, for example, as a "black box" by any other (potentially static) algorithm, while this is not generally the case for *randomized* dynamic algorithms. This motivates the search for deterministic fully dynamic approximation algorithms, even though a randomized algorithm with the same approximation guarantee might exists.

Up to date, the best answer to the first question is the randomized 2 approximation $O(\log n)$ update time algorithm from [2]. It remains elusive to design a *better-than-two* approximation factor with polylogarithmic update time. Some recent works have focused on achieving such approximation factor with lowest update time possible. The current best update time is $O(m^{1/4}/\varepsilon^{2.5})$ [3, 4], which is deterministic and guarantees a $(3/2+\varepsilon)$ approximation factor.

For the second question, deterministic polylogarithmic-time $(1+\varepsilon)$-approximation algorithms were known for the special case of low arboricity graphs [11, 12, 15]. On general graphs, the paper [5] achieved a deterministic $(3+\varepsilon)$-approximation polylogarithmic-time algorithm by maintaining a fractional matching; this algorithm however works only for the value version. *No* deterministic $o(\log n)$ approximation algorithm with polylogarithmic update time was known for the matching version. (There were many deterministic constant approximation algorithms with $o(m)$ update time for the matching version (e.g. [3–5, 7, 12]). The fastest among them requires $O(m^{1/4}/\varepsilon^{2.5})$ update time [4].)

Our Results. We make progress on both versions of the problem as stated in Theorems 1.1 and 1.2.

Theorem 1.1. *For every $\varepsilon \in (0,1)$, there is a deterministic algorithm that maintains a $(2+\varepsilon)$-approximate maximum matching in a graph in $O(poly(\log n, 1/\varepsilon))$ update time, where n denotes the number of nodes.*

[1] In this discussion, we ignore whether the update time is amortized or worst-case as this is not the focus of this paper. The update time of our algorithm is amortized.

Theorem 1.1 answers Onak and Rubinfeld's second question positively. In fact, our approximation guarantee almost matches the best (2-approximation) one provided by a randomized algorithm [2].[2] Our algorithm for Theorem 1.1 is obtained by combining previous techniques [5, 7, 15] with two new ideas that concern fractional matchings. First, we dynamize the *degree splitting process* previously used in the parallel and distributed algorithms literature [9] and use it to reduce the size of the support of the fractional matching maintained by the algorithm of [5]. This helps us maintain an approximate integral matching cheaply using the result in [7]. This idea alone already leads to a $(3+\varepsilon)$-approximation deterministic algorithm. Second, we improve the approximation guarantee further to $(2+\varepsilon)$ by proving a new structural lemma that concerns the ratio between (i) the maximum (integral) matching in the support of a maximal fractional matching and (ii) the maximum (integral) matching in the whole graph. It was known that this ratio is at least $1/3$. We can improve this ratio to $1/2$ with a fairly simple proof (using Vizing's theorem [18]). We note that this lemma can be used to improve the analysis of an algorithm in [5] to get the following result: There is a deterministic algorithm that maintains a $(2+\varepsilon)$ approximation to the size of the maximum matching in a general graph in $O(m^{1/3}/\varepsilon^2)$ amortized update time.

Theorem 1.2. *For every sufficiently large positive integral constant K, we can maintain an α_K-approximation to the value of the maximum matching[3] in a bipartite graph $G=(V,E)$, where $1 \leq \alpha_K < 2$. The algorithm is deterministic and has an amortized update time of $O(n^{2/K})$.*

We consider Theorem 1.2 to be a step towards achieving a polylogarithmic time (randomized or deterministic) fully dynamic algorithm with an approximation ratio less than 2, i.e., towards answering Onak and Rubinfeld's first question. This is because, firstly, it shows that on bipartite graphs the better-than-two approximation factor can be achieved with arbitrarily small polynomial update time, as opposed to the previous best $O(m^{1/4})$ time of [3]. Secondly, it rules out a natural form of hardness result and thus suggests that a polylogarithmic-time algorithm with better-than-two approximation factor exists on bipartite graphs. More precisely, the known hardness results (e.g. those in [1, 8, 11, 14]) that rule out a polylogarithmic-time α-approximation algorithm (for any $\alpha > 0$) are usually in the form *"assuming some conjecture, there exists a constant $\delta > 0$ such that for any constant $\varepsilon > 0$, there is no $(1-\varepsilon)\alpha$-approximation algorithm that has $n^{\delta-\varepsilon}$ update time"*; for example, for dynamically 2-*approximating* graph's diameter, this statement was proved for $\alpha = 2$ and $\delta = 1/2$ in [8], implying that any better-than-two approximation algorithm for this problem will require an update time close to $n^{1/2}$. Our result in 1.2 implies that a similar statement cannot be proved for $\alpha = 2$ for the bipartite matching problem since, for any constant $\delta > 0$, there is a $(2-\varepsilon)$-approximation algorithm with update time, say, $O(n^{\delta/2})$ for some $\varepsilon > 0$.

To derive an algorithm for Theorem 1.2, we use the fact that in a bipartite graph the size of the maximum fractional matching is the same as the size of the maximum integral matching. Accordingly, a *maximal fractional matching* (which gives 2-approximation) can be *augmented* by a fractional *b-matching*, for a carefully chosen capacity vector b, to obtain a better-than-two approximate fractional matching. The idea of "augmenting a bad solution" that we use here is inspired by the approach in the streaming setting by Konrad et al. [10]. But the way it is implemented is different as [10] focuses on using augmenting paths while we use fractional b-matchings.

Organization. In Section 1.1, we define some basic concepts and notations that will be used throughout the rest of this paper. In Section 2, we give an overview of our algorithm for Theorem 1.1. In Section 3, we highlight the main ideas behind our algorithm for Theorem 1.2. Finally, we conclude with some open problems in Section 4. All the missing details can be found in Part II and Part III of the paper.

[2] By combining our result with the techniques of [6] in a standard way, we also obtain a deterministic $(4+\varepsilon)$-approximation $O(\text{poly}\log n \text{poly}(1/\varepsilon)\log W)$-time for the dynamic maximum-weight matching problem, where W is the ratio between the largest and smallest edge weights.

[3] We can actually maintain an approximate *fractional* matching with the same performance bounds.

1.1 Notations and preliminaries

Let $n = |V|$ and $m = |E|$ respectively denote the number of nodes and edges in the input graph $G = (V, E)$. Note that m changes with time, but n remains fixed. Let $\deg_v(E')$ denote the number of edges in a subset $E' \subseteq E$ that are incident upon a node $v \in V$. An (integral) matching $M \subseteq E$ is a subset of edges that do not share any common endpoints. The *size* of a matching is the number of edges contained in it. We are also interested in the concept of a *fractional matching*. Towards this end, we first define the notion of a *fractional assignment*. A fractional assignment w assigns a weight $w(e) \geq 0$ to every edge $e \in E$. We let $W_v(w) = \sum_{(u,v)\in E} w(u,v)$ denote the total weight received by a node $v \in V$ under w from its incident edges. Further, the *support* of w is defined to be the subset of edges $e \in E$ with $w(e) > 0$. Given two fractional assignments w, w', we define their addition $(w + w')$ to be a new fractional assignment that assigns a weight $(w + w')(e) = w(e) + w'(e)$ to every edge $e \in E$. We say that a fractional assignment w forms a *fractional matching* iff we have $W_v(w) \leq 1$ for all nodes $v \in V$. Given any subset of edges $E' \subseteq E$, we define $w(E') = \sum_{e\in E'} w(e)$. We define the *size* of a fractional matching w to be $w(E)$. Given any subset of edges $E' \subseteq E$, we let $\text{Opt}_f(E')$ (resp. $\text{Opt}(E')$) denote the maximum possible size of a fractional matching with support E' (resp. the maximum possible size of an integral matching $M' \subseteq E'$). Theorem 1.3 follows from the half-integrality of the matching polytope in general graphs and its total unimodularity in bipartite graphs.

Theorem 1.3. *Consider any subset of edges $E' \subseteq E$ in the graph $G = (V, E)$. We have: $Opt(E') \leq Opt_f(E') \leq (3/2) \cdot Opt(E')$. Further, if the graph G is bipartite, then we have: $Opt_f(E') = Opt(E')$.*

Gupta and Peng [7] gave a dynamic algorithm that maintains a $(1+\varepsilon)$-approximate maximum matching in $O(\sqrt{m}/\varepsilon^2)$ update time. A simple modification of their algorithm gives the following result.

Theorem 1.4. *[7] If the maximum degree in a dynamic graph never exceeds some threshold d, then we can maintain a $(1+\varepsilon)$-approximate maximum matching in $O(d/\varepsilon^2)$ update time.*

We say that a fractional matching w is *α-maximal*, for $\alpha \geq 1$, iff $W_u(w) + W_v(w) \geq 1/\alpha$ for every edge $(u, v) \in E$. Using LP-duality and complementary slackness conditions, one can show the following result.

Lemma 1.5. *$Opt_f(E) \leq 2\alpha \cdot w(E)$ for every α-maximal fractional matching w in a graph $G = (V, E)$.*

2 General Graphs

We give a dynamic algorithm for maintaining an approximate maximum matching in a general graph. We consider the following dynamic setting. Initially, the input graph is empty. Subsequently, at each time-step, either an edge is inserted into the graph, or an already existing edge is deleted from the graph. The node-set of the graph, however, remains unchanged. Our main result in this section is stated in Theorem 1.1. Throughout this section, we will use the notations and concepts introduced in Section 1.1.

2.1 Maintaining a large fractional matching

Our algorithm for Theorem 1.1 builds upon an existing dynamic data structure that maintains a large fractional matching. This data structure was developed in [5], and can be described as follows. Fix a small constant $\varepsilon > 0$. Define $L = \lceil \log_{(1+\varepsilon)} n \rceil$, and partition the node-set V into $L+1$ subsets $V_0, \ldots, V_L$. We say that the nodes belonging to the subset V_i are in "level i". We denote the level of a node v by $\ell(v)$, i.e., $v \in V_i$ iff $\ell(v) = i$. We next define the "*level of an edge*" (u, v) to be $\ell(u, v) = \max(\ell(u), \ell(v))$. In other words, the level of an edge is the maximum level of its endpoints. We let $E_i = \{e \in E : \ell(e) = i\}$ denote the set of edges at level i, and define the subgraph $G_i = (V, E_i)$. Thus, note that the edge-set E is partitioned by the subsets $E_0, \ldots, E_L$. For each level $i \in \{0, \ldots, L\}$, we now define a fractional assignment w_i with support E_i. The fractional assignment w_i is uniform, in the sense that it assigns the same weight $w_i(e) = 1/d_i$, where $d_i = (1+\varepsilon)^i$, to every edge $e \in E_i$ in its support. In contrast, $w_i(e) = 0$ for every edge $e \in E \setminus E_i$. Throughout the rest of this section, we refer to this structure as a *"hierarchical partition"*.

Theorem 2.1. *[5] We can maintain a hierarchical partition dynamically in $O(\log n/\varepsilon^2)$ update time. The algorithm ensures that the fractional assignment $w = \sum_{i=0}^{L} w_i$ is a $(1+\varepsilon)^2$-maximal matching in $G = (V,E)$. Furthermore, the algorithm ensures that $1/(1+\varepsilon)^2 \leq W_v(w) \leq 1$ for all nodes $v \in V$ at levels $\ell(v) > 0$.*

Corollary 2.2. *The fractional matching w in Theorem 2.1 is a $2(1+\varepsilon)^2$-approximation to $Opt_f(E)$.*

Proof. Follows from Lemma 1.5 and Theorem 2.1. □

Corollary 2.3. *Consider the hierarchical partition in Theorem 2.1. There, we have $deg_v(E_i) \leq d_i$ for all nodes $v \in V$ and levels $0 \leq i \leq L$.*

Proof. The corollary holds since $1 \geq W_v(w) \geq W_v(w_i) = \sum_{(u,v)\in E_i} w_i(u,v) = (1/d_i) \cdot \deg_v(E_i)$. □

Accordingly, throughout the rest of this section, we refer to d_i as being the *degree threshold* for level i.

2.2 An overview of our approach

We will now explain the main ideas that are needed to prove Theorem 1.1. Due to space constraints, we will focus on getting a constant approximation in $O(\text{poly}\log n)$ update time. See the full version of the paper for the complete proof of Theorem 1.1. First, we maintain a hierarchical partition as per Theorem 2.1. This gives us a $2(1+\varepsilon)^2$-approximate maximum fractional matching (see Corollary 2.2). Next, we give a dynamic data structure that deterministically *rounds* this fractional matching into an integral matching without losing too much in the approximation ratio. The main challenge is to ensure that the data structure has $O(\text{poly}\log n)$ update time, for otherwise one could simply use any deterministic rounding algorithm that works well in the static setting.

2.2.1 An ideal skeleton

Our dynamic rounding procedure, when applied on top of the data structure used for Theorem 2.1, will output a low-degree subgraph that approximately preserves the size of the maximum matching. We will then extract a large integral matching from this subgraph using Theorem 1.4. To be more specific, recall that w is the fractional matching maintained in Theorem 2.1. We will maintain a subset of edges $E' \subseteq E$ in $O(\text{poly}\log n)$ update time that satisfies two properties.

$$\text{There is a fractional matching } w' \text{ with support } E' \text{ such that } w(E) \leq c \cdot w'(E') \text{ for some constant } c \geq 1. \quad (1)$$

$$\deg_v(E') = O(\text{poly}\log n) \text{ for all nodes } v \in V. \quad (2)$$

Equation 1, along with Corollary 2.2 and Theorem 1.3, guarantees that the subgraph $G' = (V,E')$ preserves the size of the maximum matching in $G = (V,E)$ within a constant factor. Equation 2, along with Theorem 1.4, guarantees that we can maintain a matching $M' \subseteq E'$ in $O(\text{poly}\log n/\varepsilon^2)$ update time such that $\text{Opt}(E') \leq (1+\varepsilon) \cdot |M'|$. Setting ε to be some small constant (say, $1/3$), these two observations together imply that we can maintain a $O(1)$-approximate maximum matching $M' \subseteq E$ in $O(\text{poly}\log n)$ update time.

To carry out this scheme, we note that in the hierarchical partition the degree thresholds $d_i = (1+\varepsilon)^i$ get smaller and smaller as we get into lower levels (see Corollary 2.3). Thus, if most of the value of $w(E)$ is coming from the lower levels (where the maximum degree is already small), then we can easily satisfy equations 1, 2. Specifically, we fix a level $0 \leq L' \leq L$ with degree threshold $d_{L'} = (1+\varepsilon)^{L'} = \Theta(\text{poly}\log n)$, and define the edge-set $Y = \bigcup_{j=0}^{L'} E_j$. We also define $w^+ = \sum_{i>L'} w_i$ and $w^- = \sum_{i\leq L'} w_i$. Note that $w(E) = w^+(E) + w^-(E)$. Now, consider two possible cases.

Case 1. $w^-(E) \geq (1/2) \cdot w(E)$. In other words, most of the value of $w(E)$ is coming from the levels $[0,L']$. By Corollary 2.3, we have $\deg_v(Y) \leq \sum_{j=0}^{L'} d_j \leq (L'+1) \cdot d_{L'} = \Theta(\text{poly}\log n)$ for all nodes $v \in V$. Thus, we can simply set $w' = w^+$ and $E' = Y$ to satisfy equations 1, 2.

Case 2. $w^+(E) > (1/2)\cdot w(E)$. In other words, most of the value of $w(E)$ is coming from the levels $[L'+1, L]$. To deal with this case, we introduce the concept of an *ideal skeleton*. See Definition 2.4. Basically, this is a subset of edges $X_i \subseteq E_i$ that scales *down* the degree of every node by a factor of $d_i/d_{L'}$. We will later show how to maintain a structure akin to an ideal skeleton in a dynamic setting. Once this is done, we can easily construct a new fractional assignment $\hat{w}_i$ that scales *up* the weights of the surviving edges in X_i by the same factor $d_i/d_{L'}$. Since $w_i(e) = 1/d_i$ for all edges $e \in E_i$, we set $\hat{w}_i(e) = 1/d_{L'}$ for all edges $e \in X_i$. To ensure that X_i is the support of the fractional assignment $\hat{w}_i$, we set $\hat{w}_i(e) = 0$ for all edges $e \in E \setminus X_i$. Let $X = \cup_{i>L'} X_i$ and $\hat{w} = \sum_{i>L'} \hat{w}_i$. It is easy to check that this transformation preserves the weight received by a node under the fractional assignment w^+, that is, we have $W_v(\hat{w}) = W_v(w^+)$ for all nodes $v \in V$. Accordingly, Lemma 2.5 implies that if we set $w' = \hat{w}$ and $E' = X$, then equations 1, 2 are satisfied.

Definition 2.4. *Consider any level $i > L'$. An* ideal skeleton *at level i is a subset of edges $X_i \subseteq E_i$ such that $deg(v, X_i) = (d_{L'}/d_i)\cdot deg(v, E_i)$ for all nodes $v \in V$. Define a fractional assignment $\hat{w}_i$ on support X_i by setting $\hat{w}_i(e) = 1/d_{L'}$ for all $e \in X_i$. For every other edge $e \in E \setminus X_i$, set $\hat{w}_i(e) = 0$. Finally, define the edge-set $X = \bigcup_{i>L'} X_i$ and the fractional assignment $\hat{w} = \sum_{j>L'} \hat{w}_j$.*

Lemma 2.5. *We have: $deg_v(X) = O(poly\log n)$ for all nodes $v \in V$, and $\hat{w}(E) = w^+(E)$. The edge-set X and the fractional assignment $\hat{w}$ are defined as per Definition 2.4.*

Proof. Fix any node $v \in V$. Corollary 2.3 and Definition 2.4 imply that: $\deg_v(X) = \sum_{j>L'} \deg_v(X_j) = \sum_{j>L'} (d_{L'}/d_j) \times \deg_v(E_j) \leq \sum_{j>L'} (d_{L'}/d_j) d_j = (L - L') d_{L'} = O(\text{poly}\log n)$.

To prove the second part, consider any level $i > L'$. Definition 2.4 implies that $W_v(\hat{w}_i) = (1/d_{L'}) \cdot \deg_v(X_i) = (1/d_i)\cdot \deg_v(E_i) = W_v(w_i)$. Accordingly, we infer that: $W_v(\hat{w}) = \sum_{i>L'} W_v(\hat{w}_i) = \sum_{i>L'} W_v(w_i) = W_v(w^+)$. Summing over all the nodes, we get: $\sum_{v\in V} W_v(\hat{w}) = \sum_{v\in V} W_v(w^+)$. It follows that $\hat{w}(E) = w^+(E)$. □

2.2.2 A degree-splitting procedure

It remains to show to how to maintain an ideal skeleton. To gain some intuition, let us first consider the problem in a static setting. Fix any level $i > L'$, and let $\lambda_i = d_i/d_{L'}$. An ideal skeleton at level i is simply a subset of edges $X_i \subseteq E_i$ that scales down the degree of every node (w.r.t. E_i) by a factor λ_i. Can we compute such a subset X_i in $O(|E_i|\cdot \text{poly}\log n)$ time? Unless we manage to solve this problem in the static setting, we cannot expect to get a dynamic data structure for the same problem with $O(\text{poly}\log n)$ update time. The SPLIT(E_i) subroutine described below answers this question in the affirmative, albeit for $\lambda_i = 2$. Specifically, in linear time the subroutine outputs a subset of edges where the degree of each node is halved. If $\lambda_i > 2$, then to get an ideal skeleton we need to repeatedly invoke this subroutine $\log_2 \lambda_i$ times: each invocation of the subroutine reduces the degree of each node by a factor of two, and hence in the final output the degree of each node is reduced by a factor of λ_i.[4] This leads to a total runtime of $O(|E_i|\cdot \log_2 \lambda_i) = O(|E_i|\cdot \log n)$ since $\lambda_i = d_i/d_{L'} \leq d_i \leq n$.

The SPLIT($\mathscr{E}$) subroutine, where $\mathscr{E} \subseteq E$. To highlight the main idea, we assume that (1) $\deg_v(\mathscr{E})$ is even for every node $v \in V$, and (2) there are an even number of edges in $\mathscr{E}$. Hence, there exists an Euler tour on $\mathscr{E}$ that visits each edge exactly once. We construct such an Euler tour in $O(|\mathscr{E}|)$ time and then collect alternating edges of this Euler tour in a set $\mathscr{H}$. It follows that (1) $\mathscr{H} \subseteq \mathscr{E}$ with $|\mathscr{H}| = |\mathscr{E}|/2$, and (2) $\deg_v(|\mathscr{H}|) = (1/2)\cdot \deg_v(|\mathscr{E}|)$ for every node $v \in V$. The subroutine returns the set of edges $\mathscr{H}$. In other words, the subroutine runs in $O(|\mathscr{E}|)$ time, and returns a subgraph that halves the degree of every node.

[4] To highlight the main idea, we assume that λ_i is a power of 2.

2.3 From ideal to approximate skeleton

We now shift our attention to maintaining an ideal skeleton in a dynamic setting. Specifically, we focus on the following problem: We are given an input graph $G_i = (V, E_i)$, with $|V| = n$, that is undergoing a sequence of edge insertions/deletions. The set E_i corresponds to the level i edges in the hierarchical partition (see Section 2.1). We always have $\deg_v(E_i) \leq d_i$ for all nodes $v \in V$ (see Corollary 2.3). There is a parameter $1 \leq \lambda_i = d_i/d_{L'} \leq n$. In $O(\text{poly}\log n)$ update time, we want to maintain a subset of edges $X_i \subseteq E_i$ such that $\deg_v(X_i) = (1/\lambda_i) \cdot \deg_v(E_i)$ for all nodes $v \in V$. The basic building block of our dynamic algorithm will be the (static) subroutine SPLIT($\mathscr{E}$) from Section 2.2.2.

Unfortunately, we will not be able to achieve our initial goal, which was to reduce the degree of *every* node by *exactly* the factor λ_i in a dynamic setting. For one thing, there might be some nodes v with $\deg_v(E_i) < \lambda_i$. It is clearly not possible to reduce their degrees by a factor of λ_i (otherwise their new degrees will be between zero and one). Further, we will need to introduce some *slack* in our data structures if we want to ensure polylogarithmic update time.

We now describe the structures that will be actually maintained by our dynamic algorithm. We maintain a partition of the node-set V into two subsets: $B_i \subseteq V$ and $T_i = V \setminus B$. The nodes in B_i (resp. T_i) are called "big" (resp. "tiny"). We also maintain a subset of nodes $S_i \subseteq V$ that are called "spurious". Finally, we maintain a subset of edges $X_i \subseteq E_i$. Fix two parameters $\varepsilon, \delta \in (0,1)$. For technical reasons that will become clear later on, we require that:

$$\varepsilon = 1/100, \text{ and } \delta = \varepsilon^2/L \tag{3}$$

We ensure that the following properties are satisfied.

$$\deg_v(E_i) \geq \varepsilon d_i/L \text{ for all nodes } v \in B_i \setminus S_i. \tag{4}$$

$$\deg_v(E_i) \leq 2\varepsilon d_i/L \text{ for all nodes } v \in T_i \setminus S_i. \tag{5}$$

$$|S_i| \leq \delta \cdot |B_i| \tag{6}$$

$$\frac{(1-\varepsilon)}{\lambda_i} \cdot \deg_v(E_i) \leq \deg_v(X_i) \leq \frac{(1+\varepsilon)}{\lambda_i} \cdot \deg_v(E_i) \text{ for all nodes } v \in B_i \setminus S_i. \tag{7}$$

$$\deg_v(X_i) \leq (1/\lambda_i) \cdot (2\varepsilon d_i/L) \text{ for all nodes } v \in T_i \setminus S_i. \tag{8}$$

$$\deg_v(X_i) \leq (1/\lambda_i) \cdot d_i \text{ for all nodes } v \in S_i. \tag{9}$$

Equation 4 implies that all the non-spurious big nodes have large degrees in $G_i = (V, E_i)$. On a similar note, equation 5 implies that all the non-spurious tiny nodes have small degrees in G_i. Next, by equation 6, the number of spurious nodes is negligibly small in comparison with the number of big nodes. By equation 7, the degrees of the non-spurious big nodes are scaled by a factor that is very close to λ_i. Thus, the non-spurious big nodes satisfy an approximate version of the degree-splitting property required by Definition 2.4.

Moving on, by equation 8, the degrees of the non-spurious tiny nodes in X_i are at most $(1/\lambda_i) \cdot (2\varepsilon d_i/L) = 2\varepsilon d_{L'}/L$. Since each edge in X_i receives weight $1/d_{L'}$ under the assignment $\hat{w}_i$ (see Definition 2.4), we infer that $W_v(\hat{w}_i) = (1/d_{L'}) \cdot \deg_v(X_i) \leq 2\varepsilon/L$ for all nodes $v \in T_i \setminus S_i$. Since there are at most $(L - L')$ relevant levels in a hierarchical partition, we infer that:

$$\sum_{i > L' : v \in T_i \setminus S_i} W_v(\hat{w}_i) \leq L \cdot (2\varepsilon/L) = 2\varepsilon \tag{10}$$

Since for a non-spurious tiny node $v \in T_i \setminus S_i$ we have $\deg_v(E_i) \leq 2\varepsilon d_i/L$ (see equation 5) and $W_v(w_i) = (1/d_i) \cdot \deg_v(E_i) \leq 2\varepsilon/L$, an exactly similar argument gives us:

$$\sum_{i > L' : v \in T_i \setminus S_i} W_v(w_i) \leq L \cdot (2\varepsilon/L) = 2\varepsilon \tag{11}$$

Equations 10, 11 have the following implication: The levels where v is a non-spurious tiny node contribute a negligible amount towards the weights $W_v(w^+)$ and $W_v(\hat{w})$ (see Section 2.2.1). Hence, although we are no longer guaranteed that the degrees of these nodes will be scaled down exactly by the factor λ_i, this should not cause too much of a problem – the sizes of the fractional assignments $w^+(E)$ and $\hat{w}(E)$ should still be close to each other as in Section 2.2.1.

Finally, Corollary 2.3 states that the degree of a node in E_i is at most d_i. Hence, according to the definition of an ideal skeleton (see Definition 2.4), the degree of a node in X_i ought not to exceed $(1/\lambda_i) \cdot d_i = d_{L'}$. Equation 9 ensures that the spurious nodes satisfy this property.

If the set of edges X_i satisfies the conditions described above, then we say that we have an *approximate-skeleton* at our disposal. This is formally stated as follows.

Definition 2.6. *Fix any level $i > L'$, and suppose that there is a partition of the node-set V into two subsets $B_i \subseteq V$ and $T_i = V \setminus B_i$. Further, consider another subset of nodes $S_i \subseteq V$ and a subset of edges $X_i \subseteq E_i$. The tuple (B_i, T_i, S_i, X_i) is an* approximate-skeleton *iff it satisfies equations (4) – (9).*

One may object at this point that we have deviated from the concept of an ideal skeleton (see Definition 2.4) so much that it will impact the approximation ratio of our final algorithm. To address this concern, we now state the following theorem whose proof appears in Section 2.5.

Theorem 2.7. *For each level $i > L'$, consider an approximate skeleton as per Definition 2.6. Let $X = \bigcup_{i>L'} X_i$ denote the set of edges from these approximate-skeletons. Let $Y = \bigcup_{i \leq L'} E_i$ denote the set of edges from the remaining levels in the hierarchical partition. Then we have:*

1. *There is a fractional matching w' on support $X \cup Y$ such that $w(E) \leq O(1) \cdot w'(X \cup Y)$. Here, w is the fractional matching given by Theorem 2.1.*
2. *$deg_v(X \cup Y) = O(poly \log n)$ for all $v \in V$.*

In other words, the set of edges $X \cup Y$ satisfies equations 1, 2.

As per the discussion immediately after equations 1, 2, we infer the following guarantee.

Corollary 2.8. *Suppose that for each level $i > L'$ there is a dynamic algorithm that maintains an approximate-skeleton in $O(poly \log n)$ update time. Then we can also maintain a $O(1)$-approximate maximum matching in the input graph G in $O(poly \log n)$ update time.*

It remains to show how to maintain an approximate skeleton efficiently in a dynamic setting. Accordingly, we state the following theorem whose proof appears in Section 2.4.

Theorem 2.9. *Consider any level $i > L'$. In $O(poly \log n)$ update time, we can maintain an approximate-skeleton at level i as per Definition 2.6.*

Corollary 2.8 and Theorem 2.9 imply that we can maintain a $O(1)$-approximate maximum matching in a dynamic graph in $O(\text{poly} \log n)$ update time.

2.4 Maintaing an approximate skeleton: Proof of Theorem 2.9

Fix a level $i > L'$. We will show how to efficiently maintain an approximate skeleton at level i under the assumption that $\lambda_i = 2$. In the full version of the paper, if $\lambda_i > 2$, then we iteratively apply the algorithm presented here $O(\log_2 \lambda_i) = O(\log_2(d_i/d_{L'})) = O(\log n)$ times, and each iteration reduces the degrees of the nodes by a factor of two. Hence, after the final iteration, we get a subgraph that is an approximate skeleton as per Definition 2.6.

We maintain the set of edges $E_{B_i} = \{(u,v) \in E_i : \{u,v\} \cap B_i \neq \emptyset\}$ that are incident upon the big nodes. Further, we associate a "status bit" with each node $v \in V$, denoted by $\textsc{Status}[v] \in \{0,1\}$. We ensure that they satisfy two conditions: (1) If $\textsc{Status}[v] = 1$, then $\deg_v(E_i) \geq \varepsilon d_i/L$ (which is the threshold for non-spurious big nodes in equation 4). (2) If $\textsc{Status}[v] = 0$, then $\deg_v(E_i) \leq 2\varepsilon d_i/L$ (which is the threshold for

non-spurious tiny nodes in equation 5). Whenever an edge incident upon v is inserted into (resp. deleted from) E_i, we update the status bit of v *in a lazy manner* (i.e., we flip the bit only if one of the two conditions is violated). We define an "epoch" of a node v to be the time-interval between any two consecutive flips of the bit STATUS$[v]$. Since there is a gap of $\varepsilon d_i/L$ between the thresholds in equations 4, 5, we infer that:

In any epoch of a node v, at least $\varepsilon d_i/L$ edge insertions/deletions incident upon v takes place in E_i. (12)

Our dynamic algorithm runs in *"phases"*. In the beginning of a phase, there are no spurious nodes, i.e., we have $S_i = \emptyset$. During a phase, we handle the insertion/deletion of an edge in E_i as follows.

2.4.1 Handling the insertion/deletion of an edge

Consider the insertion/deletion of an edge (u,v) in E_i. To handle this event, we first update the set E_i and the status bits of u,v. If $\{u,v\} \cap B_i \neq \emptyset$, then we also update the edge-set E_{B_i}. Next, for every endpoint $x \in \{u,v\} \setminus S_i$, we check if the node x violates any of the equations 4, 5, 7. If yes, then we set $S_i \leftarrow S_i \cup \{x\}$. Finally, we check if $|S_i| > \delta \cdot |B_i|$, and if yes, then we terminate the phase by calling the subroutine TERMINATE-PHASE(.).

2.4.2 The subroutine TERMINATE-PHASE(.)

We scan through the nodes in S_i. For each such node $v \in S_i$, if STATUS$[v] = 1$, then we set $B_i \leftarrow B_i \cup \{v\}$, $T_i \leftarrow T_i \setminus \{v\}$, and ensure that all the edges $(u,v) \in E_i$ incident upon v are included in E_{B_i}. Else if STATUS$[v] = 0$, then we set $T_i \leftarrow T_i \cup \{v\}$, $B_i \leftarrow B_i \setminus \{v\}$, and ensure that all the edges $(u,v) \in E_i$ incident upon v are excluded from E_{B_i}. Finally, we set $S_i \leftarrow \emptyset$ and $X_i \leftarrow$ SPLIT(E_{B_i}) (see Section 2.2.2). From the next edge insertion/deletion in E_i, we begin a new phase.

2.4.3 Correctness.

At the start of a phase, clearly all the properties hold. This fact, along with the observation that an edge is never inserted into X_i during the middle of a phase, implies that equations 8, 9 hold all the time. Whenever a node violates equations 4, 5, 7, we make it spurious. Finally, whenever equation 6 is violated, we terminate the phase. This ensures that all the properties hold all the time.

2.4.4 Analyzing the amortized update time.

Handling an edge insertion/deletion in E_i in the middle of a phase needs $O(1)$ update time. Just before a phase ends, let b and s respectively denote the number of big and spurious nodes. Since a phase ends only when equation 6 is violated, we have $s \geq \delta \cdot b$. In the subroutine TERMINATE-PHASE(.), updating the edge-set E_{B_i} requires $O(s \cdot d_i)$ time, since we need to go through all the s nodes in S, and for each such node, we need to check all the edges incident upon it (and a node can have at most d_i edges incident upon it by Corollary 2.3). At this stage, the set B_i consists of at most $(s+b)$ nodes, and so the set E_{B_i} consists of at most $(s+b)d_i$ edges. Hence, the call to the subroutine SPLIT(E_{B_i}) takes $O((s+b)d_i)$ time. Accordingly, the total time taken to terminate the phase is $O((s+b)d_i) = O((s+s/\delta)d_i) = O(sd_i/\delta)$. We thus reach the following conclusion: The total time spent on a given phase is equal to $O(sd_i/\delta)$, where s is the number of spurious nodes at the end of the phase. Since $S_i = \emptyset$ in the beginning of the phase, we can also interpret s as being the number of nodes that *becomes spurious* during the phase. Let C denote a counter that is initially set to zero, and is incremented by one each time some node becomes spurious. From the preceding discussion, it follows that the total update time of our algorithm, across all the phases, is at most $O(Cd_i/\delta)$. Let t be the total number of edge insertions/deletions in E_i. We will show that $C = O(tL/(\varepsilon^2 d_i))$. This will imply an amortized update time of $O((1/t) \cdot Cd_i/\delta) = O(L/\varepsilon^2\delta) = O(\text{poly}\log n)$. The last equality holds due to equation 3.

Note that during a phase a node v becomes spurious because of one of two reasons: (1) It violated equations 4 or 5. In this case, the node's status bit is flipped. Hence, by equation 12, between any two

such events, at least $\varepsilon d_i/L$ edge insertions/deletions occur incident upon v. (2) It violates equation 7. In this event, note that in the beginning of the phase we had $v \in B_i$, $\deg_v(X_i) = (1/2) \cdot \deg_v(E_i) = (1/\lambda_i) \cdot \deg_v(E_i)$ and $\deg_v(E_i) \geq \varepsilon d_i/L$. The former guarantee holds since we set $X_i \leftarrow \text{SPLIT}(E_{B_i})$ at the end of the previous phase, whereas the latter guarantee follows from equation 4. On the other hand, when the node v violates equation 7, we find that $\deg_v(X_i)$ differs from $(1/\lambda_i) \cdot \deg_v(E_i)$ by at least $(\varepsilon/\lambda_i) \cdot \deg_v(E_i) = (\varepsilon/2) \cdot \deg_v(E_i)$. Accordingly, during this time-interval (that starts at the beginning of the phase and ends when equation 7 is violated), at least $\Omega(\varepsilon^2 d_i/L)$ edge insertions/deletions incident upon v must have taken place in E_i. To summarize, for each unit increment in C, we must have $\Omega(\varepsilon^2 d_i/L)$ edge insertions/deletions in E_i. Thus, we have $C = O(t/(\varepsilon^2 d_i/L)) = O(tL/(\varepsilon^2 d_i))$.

2.5 Approximation guarantee from approximate skeletons: Proof of Theorem 2.7

We devote this section to the complete proof of Theorem 2.7. At a high level, the main idea behind the proof remains the same as in Section 2.2.1. We will have to overcome several intricate obstacles, however, because now we are dealing with the relaxed notion of an approximate skeleton as defined in Section 2.3.

We start by focussing on the second part of Theorem 2.7, which states that the degree of every node in $X \cup Y$ is at most $O(\text{poly}\log n)$. This is stated and proved in Lemma 2.10.

Lemma 2.10. *Consider the subsets of edges $X \subseteq E$ and $Y \subseteq E$ as per Theorem 2.7. Then we have $\deg_v(X \cup Y) = O(\text{poly}\log n)$ for every node $v \in V$.*

Proof. We first bound the degree of a node $v \in V$ in X. Towards this end, consider any level $i > L'$. By equation 9, we have that $\deg_v(X_i) \leq (1/\lambda_i) \cdot d_i = d_{L'}$ for all spurious nodes $v \in S_i$. By equation 8, we have $\deg_v(X_i) \leq (1/\lambda_i) \cdot (2\varepsilon d_i/L) = 2\varepsilon d_{L'}/L \leq d_{L'}$ for all non-spurious tiny nodes $v \in T_i \setminus S_i$. Finally, by equation 7 and Corollary 2.3, we have that $\deg_v(X_i) \leq ((1+\varepsilon)/\lambda_i) \cdot \deg_v(E_i) \leq (1+\varepsilon) \cdot (d_i/\lambda_i) = (1+\varepsilon) \cdot d_{L'}$ for all non-spurious big nodes $v \in B_i \setminus S_i$. By Definition 2.6, a node belongs to exactly one of the three subsets – S_i, $T_i \setminus S_i$ and $B_i \setminus S_i$. Combining all these observations, we get: $\deg_v(X_i) \leq (1+\varepsilon) \cdot d_{L'} = O(\text{poly}\log n)$ for all nodes $v \in V$. Now, summing over all $i > L'$, we get: $\deg_v(X) = \sum_{i>L'} \deg_v(X_i) \leq (L - L') \cdot O(\text{poly}\log n) = O(\text{poly}\log n)$ for all the nodes $v \in V$.

Next, we bound the degree of a node $v \in V$ in Y. Note that the degree thresholds in the levels $[0, L']$ are all at most $d_{L'}$. Specifically, for all $i \leq L'$ and $v \in V$, Corollary 2.3 implies that $\deg_v(E_i) \leq d_i \leq d_{L'} = O(\text{poly}\log n)$. Hence, for every node $v \in V$, we have $\deg_v(Y) = \sum_{i \leq L'} \deg_v(E_i) \leq (L'+1) \cdot O(\text{poly}\log n) = O(\text{poly}\log n)$.

To summarize, the maximum degree of a node in the edge-sets X and Y is $O(\text{poly}\log n)$. Hence, for every node $v \in V$, we have: $\deg_v(X \cup Y) = \deg_v(X) + \deg_v(Y) = O(\text{poly}\log n)$. This concludes the proof of the lemma. □

We now focus on the first part of Theorem 2.7, which guarantees the existence of a large fractional matching with support $X \cup Y$. This is stated in the lemma below. Note that Lemma 2.10 and Lemma 2.11 together imply Theorem 2.7.

Lemma 2.11. *Consider the subsets of edges $X \subseteq E$ and $Y \subseteq E$ as per Theorem 2.7. Then there exists a fractional matching w' on support $X \cup Y$ such that $w(E) \leq O(1) \cdot w'(E)$. Here, w is the fractional matching given by Theorem 2.1.*

We devote the rest of this section to the proof of Lemma 2.11. As in Section 2.2.1, we start by defining two fractional assignments $w^+ = \sum_{i>L'} w_i$ and $w^- = \sum_{i \leq L'} w_i$. In other words, w^+ captures the fractional weights assigned to the edges in levels $[L'+1, L]$ by the hierarchical partition, whereas w^- captures the fractional weights assigned to the edges in the remaining levels $[0, L']$. The fractional assignment w^+ has support $\cup_{i>L'} E_i$, whereas the fractional assignment w^- has support $\cup_{i \leq L'} E_i = Y$. We have $w = w^+ + w^-$ and $w(E) = w^+(E) + w^-(E)$. If at least half of the value of $w(E)$ is coming from the levels $[0, L']$, then

there is nothing to prove. Specifically, suppose that $w^-(E) \geq (1/2) \cdot w(E)$. Then we can set $w' = w^-$ and obtain $w(E) \leq 2 \cdot w^-(E) = 2 \cdot w^-(Y) = 2 \cdot w'(Y) = 2 \cdot w'(X \cup Y)$. This concludes the proof of the lemma. Accordingly, from this point onward, we will assume that at least half of the value of $w(E)$ is coming from the levels $i > L'$. Specifically, we have:

$$w(E) \leq 2 \cdot w^+(E) \tag{13}$$

Given equation 13, we will construct a fractional matching with support X whose size is within a constant factor of $w(E)$.[5] We want to follow the argument applied to ideal skeletons in Section 2.2.1 (see Definition 2.4). Accordingly, for every level $i > L'$ we now define a fractional assignment $\hat{w}_i$ with support X_i.

$$\begin{aligned} \hat{w}_i(e) &= 1/d_{L'} \text{ for all edges } e \in X_i \\ &= 0 \quad \text{for all edges } e \in E \setminus X_i. \end{aligned} \tag{14}$$

We next define the fractional assignment $\hat{w}$.

$$\hat{w} = \sum_{i > L'} \hat{w}_i \tag{15}$$

In Section 2.2.1 (see Lemma 2.5), we observed that $\hat{w}$ is a fractional matching with support X whose size is exactly the same as $w^+(E)$. This observation, along with equation 13, would have sufficed to prove Lemma 2.11. The intuition was that at every level $i > L'$, the degree of a node $v \in V$ in X_i is exactly $(1/\lambda_i)$ times its degree in E_i. In contrast, the weight of an edge $e \in X_i$ under $\hat{w}_i$ is exactly λ_i times its weight under w_i. This ensured that the weight of node remained unchanged as we transitioned from w_i to $\hat{w}_i$, that is, $W_v(w_i) = W_v(\hat{w}_i)$ for all nodes $v \in V$. Unfortunately, this guarantee no longer holds for approximate-skeletons. It still seems natural, however, to compare the weights a node receives under these two fractional assignments w_i and $\hat{w}_i$. This depends on the status of the node under consideration, depending on whether the node belongs to the set $B_i \setminus S_i$, $T_i \setminus S_i$ or S_i (see Definition 2.6). Towards this end, we derive Claims 2.12, 2.13, 2.14. The first claim states that the weight of a non-spurious big node under $\hat{w}_i$ is *close* to its weight under w_i. The second claim states that the weight of a non-spurious tiny node under $\hat{w}_i$ is *very small* (less than $2\varepsilon/L$). The third claim states that the weight of a spurious node under $\hat{w}_i$ is at most one.

Claim 2.12. *For all $i > L'$ and $v \in B_i \setminus S_i$, we have:*

$$(1-\varepsilon) \cdot W_v(w_i) \leq W_v(\hat{w}_i) \leq (1+\varepsilon) \cdot W_v(w_i).$$

Proof. Fix any level $i > L'$ and any node $v \in B_i \setminus S_i$. The claim follows from equation 7 and the facts below:

(1) $\lambda_i = d_i/d_{L'}$.

(2) $W_v(\hat{w}_i) = (1/d_{L'}) \cdot \deg_v(X_i)$. See equation 14.

(3) $W_v(w_i) = (1/d_i) \cdot \deg_v(E_i)$. See Section 2.1. □

Claim 2.13. *For all levels $i > L'$ and non-spurious tiny nodes $v \in T_i \setminus S_i$, we have $W_v(\hat{w}_i) \leq 2\varepsilon/L$.*

Proof. Fix any level $i > L'$ and any node $v \in T_i \setminus S_i$. The claim follows from equation 8 and the facts below:

(1) $\lambda_i = d_i/d_{L'}$.

(2) $W_v(\hat{w}_i) = (1/d_{L'}) \cdot \deg_v(X_i)$. See equation 14. □

Claim 2.14. *For all $i > L'$ and $v \in S_i$, we have $W_v(\hat{w}_i) \leq 1$.*

[5] Recall that w is the fractional matching given by the hierarchical partition. See Section 2.1.

Proof. Fix any level $i > L'$ and any node $v \in S_i$. The claim follows from equation 9 and the facts below:

(1) $\lambda_i = d_i/d_{L'}$.

(2) $W_v(\hat{w}_i) = (1/d_{L'}) \cdot \deg_v(X_i)$. See equation 14. □

Unfortunately, the fractional assignment $\hat{w}$ need not necessarily be a fractional matching, the main reason being that at a level $i > L'$ the new weight $W_v(\hat{w}_i)$ of a spurious node $v \in S_i$ can be much larger than its original weight $W_v(w_i)$. Specifically, Claim 2.14 permits that $W_v(\hat{w}_i) = 1$ for such a node $v \in S_i$. If there exists a node $v \in V$ that belongs to S_i at every level $i > L'$, then we might have $W_v(\hat{w}) = \sum_{i>L'} W_v(\hat{w}_i) = \sum_{i>L'} 1 = (L - L') >> 1 \geq W_v(w)$.

To address this concern regarding the weights of the spurious nodes, we switch from $\hat{w}$ to a new fractional assignment w'', which is defined as follows. For every level $i > L'$, we construct a fractional assignment w''_i that sets to zero the weight of every edge in X_i that is incident upon a spurious node $v \in S_i$. For every other edge e, the weight $w''_i(e)$ remains the same as $\hat{w}_i(e)$. Then we set $w'' = \sum_{i>L'} w''_i$.

$$\begin{aligned} w''_i(u,v) &= \hat{w}_i(u,v) \text{ if } (u,v) \in X_i \text{ and } \{u,v\} \cap S_i = \emptyset \\ &= 0 \text{ if } (u,v) \in X_i \text{ and } \{u,v\} \cap S_i \neq \emptyset. \\ &= 0 \text{ else if } (u,v) \in E \setminus X_i \end{aligned} \tag{16}$$

$$w'' = \sum_{i>L'} w''_i \tag{17}$$

The above transformation guarantees that $W_v(w''_i) = 0$ for every spurious node $v \in S_i$ at level $i > L'$. Thus, the objection raised above regarding the weights of spurious nodes is no longer valid for the fractional assignment w''_i. We now make three claims on the fractional assignments $\hat{w}$ and w''.

Claim 2.15 bounds the maximum weight of a node under w''. Its proof appears in Section 2.5.1.

Claim 2.15. *We have $W_v(w'') \leq 1 + 3\varepsilon$ for all $v \in V$.*

Claim 2.16 states that the size of w'' is close to the size of $\hat{w}$. Its proof appears in Section 2.5.2.

Claim 2.16. *We have $w''(E) \geq \hat{w}(E) - 4\varepsilon \cdot w^+(E)$.*

Claim 2.17 states that the size of $\hat{w}$ is within a constant factor of the size of w^+. Its proof appears in Section 2.5.3.

Claim 2.17. *We have $\hat{w}(E) \geq (1/8) \cdot w^+(E)$.*

Corollary 2.18. *We have $w''(E) \geq (1/8 - 4\varepsilon) \cdot w^+(E)$.*

Proof. Follows from Claims 2.16 and 2.17. □

To complete the proof of Lemma 2.11, we scale down the weights of the edges in w'' by a factor of $(1+3\varepsilon)$. Specifically, we define a fractional assignment w' such that:

$$w'(e) = \frac{w''(e)}{(1+3\varepsilon)} \text{ for all edges } e \in E.$$

Since w'' has support X, the fractional assignment w' also has support X, that is, $w'(e) = 0$ for all edges $e \in E \setminus X$. Claim 2.15 implies that $W_v(w') = W_v(w'')/(1+3\varepsilon) \leq 1$ for all nodes $v \in V$. Thus, w' is fractional matching on support X. Since the edge-weights are scaled down by a factor of $(1+3\varepsilon)$, Corollary 2.18 implies that:

$$w'(E) = \frac{w''(E)}{(1+3\varepsilon)} \geq \frac{(1/8 - 4\varepsilon)}{(1+3\varepsilon)} \cdot w^+(E). \tag{18}$$

Equations 13 and 18 imply that $w(E) \leq O(1) \cdot w'(E)$. This concludes the proof of Lemma 2.11.

2.5.1 Proof of Claim 2.15

Throughout the proof, we fix any given node $v \in V$. We will show that $W_v(w'') \leq 1+3\varepsilon$. We start by making a simple observation:

$$W_v(w''_i) \leq W_v(\hat{w}_i) \text{ for all levels } i > L'. \tag{19}$$

Equation 19 holds since we get the fractional assignment w''_i from $\hat{w}_i$ by setting some edge-weights to zero and keeping the remaining edge-weights unchanged (see equation 16).

By Definition 2.6, at every level $i > L'$ the node v is part of exactly one of the three subsets – $T_i \setminus S_i$, $B_i \setminus S_i$ and S_i. Accordingly, we can classify the levels into three types depending on which of these subsets v belongs to at that level. Further, recall that $W_v(w'') = \sum_{i>L'} W_v(w''_i)$. We will separately bound the contributions from each type of levels towards the node-weight $W_v(w'')$.

We first bound the contribution towards $W_v(w'')$ from all the levels $i > L'$ where $v \in T_i \setminus S_i$.

Claim 2.19. *We have:*

$$\sum_{i>L':v\in T_i\setminus S_i} W_v(w''_i) \leq 2\varepsilon.$$

Proof. Claim 2.13 implies that:

$$\sum_{i>L':v\in T_i\setminus S_i} W_v(\hat{w}_i) \leq \sum_{i>L':v\in T_i\setminus S_i} (2\varepsilon/L) \leq 2\varepsilon. \tag{20}$$

The claim follows from equations 19 and 20. □

We next bound the contribution towards $W_v(w'')$ from all the levels $i > L'$ where $v \in B_i \setminus S_i$.

Claim 2.20. *We have:*

$$\sum_{i>L':v\in B_i\setminus S_i} W_v(w''_i) \leq 1+\varepsilon.$$

Proof. Let LHS $= \sum_{i>L':v\in B_i\setminus S_i} W_v(w''_i)$. We have:

$$\begin{aligned}
\text{LHS} &\leq \sum_{i>L':v\in B_i\setminus S_i} W_v(\hat{w}_i) && (21)\\
&\leq \sum_{i>L':v\in B_i\setminus S_i} (1+\varepsilon)\cdot W_v(w_i) && (22)\\
&\leq (1+\varepsilon)\cdot \sum_{i=0}^{L} W_v(w_i) = (1+\varepsilon)\cdot W_v(w)\\
&\leq (1+\varepsilon) && (23)
\end{aligned}$$

Equation 21 holds because of equation 19. Equation 22 follows from Claim 2.12. Finally, equation 23 holds since w is a fractional matching (see Section 2.1). □

Finally, we bound the contribution towards $W_v(w'')$ from all the levels $i > L'$ where $v \in S_i$.

Claim 2.21. *We have:*

$$\sum_{i>L':v\in S_i} W_v(w''_i) = 0.$$

Proof. Consider any level $i > L'$ where $v \in S_i$. By equation 16, every edge in X_i incident upon v has zero weight under w''_i, and hence $W_v(w''_i) = 0$. The claim follows. □

Adding up the bounds given by Claims 2.19, 2.20 and 2.21, we get:

$$\begin{aligned} W_v(w'') &= \sum_{i>L'} W_v(w''_i) \\ &= \sum_{i>L':v\in T_i\setminus S_i} W_v(w''_i) + \sum_{i>L':v\in B_i\setminus S_i} W_v(w''_i) \\ &\quad + \sum_{i>L':v\in S_i} W_v(w''_i) \\ &\leq 2\varepsilon + (1+\varepsilon) + 0 = 1+3\varepsilon. \end{aligned}$$

This concludes the proof of Claim 2.15.

2.5.2 Proof of Claim 2.16

For any given fractional assignment, the some of the node-weights is two times the sum of the edge-weights (since each edge has two endpoints). Keeping this in mind, instead of relating the sum of the edge-weights under the fractional assignments $w'', \hat{w}$ and w^+ as stated in Claim 2.16, we will attempt to relate the sum of the node-weights under w'', $\hat{w}$ and w^+.

As we switch from the fractional assignment $\hat{w}_i$ to the fractional assignment w''_i at some level $i > L'$, all we do is to set to zero the weight of any edge incident upon a spurious node in S_i. Hence, intuitively, the difference between the sum of the node-weights under $w'' = \sum_{i>L'} w''_i$ and $\hat{w} = \sum_{i>L'} \hat{w}_i$ should be bounded by the sum of the weights of the spurious nodes across all the levels $i > L'$. This is formally stated in the claim below.

Claim 2.22. *We have:*

$$\sum_{v\in V} W_v(w'') \geq \sum_{v\in V} W_v(\hat{w}) - \sum_{i>L'} \sum_{v\in S_i} 2\cdot W_v(\hat{w}_i).$$

Proof. The left hand side (LHS) of the inequality is exactly equal to two times the sum of the edge-weights under w''. Similarly, the first sum in the right hand side (RHS) is exactly equal to two times the sum of the edge-weights under $\hat{w}$. Finally, we can also express the second sum in the RHS as the sum of certain edge-weights under $\hat{w}$.

Consider any edge $(x,y) \in E$. We will show that the contribution of this edge towards the LHS is at least its contribution towards the RHS, thereby proving the claim.

Case 1. $(x,y) \notin X_i$ for all $i > L'$. Then the edge (x,y) contributes zero to the left hand side (LHS) and zero to the right hand side (RHS).

Case 2. $(x,y) \in X_i$ at some level $i > L'$, but none of the endpoints of the edge is spurious, that is, $\{x,y\} \cap S_i = \emptyset$. In this case, by equation 16, the edge (x,y) contributes $2\cdot w''_i(x,y)$ to the LHS, $2\cdot\hat{w}_i(x,y)$ to the first sum in the RHS, and zero to the second sum in the RHS. Further, we have $w''_i(x,y) = \hat{w}_i(x,y)$. Hence, the edge makes exactly the same contribution towards the LHS and the RHS.

Case 3. $(x,y) \in X_i$ at some level $i > L'$, and at least one endpoint of the edge is spurious, that is, $\{x,y\} \cap S_i \neq \emptyset$. In this case, by equation 16, the edge (x,y) contributes zero to the LHS, $2\cdot\hat{w}(x,y)$ to the first sum in the RHS, and at least $2\cdot\hat{w}(x,y)$ to the second sum in the RHS. Hence, the net contribution towards the RHS is at most zero. In other words, the contribution towards the LHS is at least the contribution towards the RHS. □

At every level $i > L'$, we will now bound the sum of the weights of the spurious nodes $v \in S_i$ under $\hat{w}$ by the sum of the node-weights under w_i. We will use the fact that each spurious node gets weight at most one (see Claim 2.14), which implies that $\sum_{v\in S_i} W_v(\hat{w}_i) \leq |S_i|$. By equation 6, we will upper bound the number of spurious nodes by the number of non-spurious big nodes. Finally, by equation 7, we will infer that each

non-spurious big node has sufficiently large degree in E_i, and hence its weight under w_i is also sufficiently large.

Claim 2.23. *For every level $i > L'$, we have:*

$$\sum_{v \in S_i} W_v(\hat{w}_i) \leq (2\delta L/\varepsilon) \cdot \sum_{v \in V} W_v(w_i).$$

Proof. Fix any level $i > L'$. Claim 2.14 states that $W_v(\hat{w}_i) \leq 1$ for all nodes $v \in S_i$. Hence, we get:

$$\sum_{v \in S_i} W_v(\hat{w}_i) \leq |S_i| \tag{24}$$

Equation 6 implies that $|S_i| \leq \delta \cdot |B_i| \leq \delta \cdot (|B_i \setminus S_i| + |S_i|)$. Rearranging the terms, we get: $|S_i| \leq \frac{\delta}{1-\delta} \cdot |B_i \setminus S_i|$. Since $\delta < 1/2$ (see equation 3), we have:

$$|S_i| \leq 2\delta \cdot |B_i \setminus S_i| \tag{25}$$

From equations 24 and 25, we get:

$$\sum_{v \in S_i} W_v(\hat{w}_i) \leq 2\delta \cdot |B_i \setminus S_i| \tag{26}$$

Now, equation 4 states that $\deg_v(E_i) \geq (\varepsilon d_i/L)$ for all nodes $v \in B_i \setminus S_i$. Further, in the hierarchical partition we have $W_v(w_i) = (1/d_i) \cdot \deg_v(E_i)$ for all nodes $v \in V$ (see Section 2.1). Combining these two observations, we get: $W_v(w_i) \geq \varepsilon/L$ for all nodes $v \in B_i \setminus S_i$. Summing over all nodes $v \in V$, we get:

$$\sum_{v \in V} W_v(w_i) \geq \sum_{v \in B_i \setminus S_i} W_v(w_i) \geq (\varepsilon/L) \cdot |B_i \setminus S_i| \tag{27}$$

The claim follows from equations 26 and 27. □

Corollary 2.24. *We have:*

$$\sum_{i > L'} \sum_{v \in S_i} W_v(\hat{w}_i) \leq (2\delta L/\varepsilon) \cdot \sum_{v \in V} W_v(w^+).$$

Proof. Follows from summing Claim 2.23 over all levels $i > L'$, and noting that since $w^+ = \sum_{i > L'} w_i$, we have $W_v(w^+) = \sum_{i > L'} W_v(w_i)$ for all nodes $v \in V$. □

From Claim 2.22 and Corollary 2.24, we get:

$$\sum_{v \in V} W_v(w'') \geq \sum_{v \in V} W_v(\hat{w}) - (4\delta L/\varepsilon) \cdot \sum_{v \in V} W_v(w^+) \tag{28}$$

Since $\delta = \varepsilon^2/L$ (see equation 3) and since the sum of the node-weights in a fractional assignment is exactly two times the sum of the edge-weights, Claim 2.16 follows from equation 28.

2.5.3 Proof of Claim 2.17

Every edge $(u, v) \in X = \cup_{i > L'} X_i$ has at least one endpoint at a level $i > L'$ (see Definition 2.6). In other words, every edge in X has at least one endpoint in the set V^* as defined below.

Definition 2.25. *Define $V^* = \{v \in V : \ell(v) > L'\}$ to be the set of all nodes at levels strictly greater than L'.*

Thus, under any given fractional assignment, the sum of the node-weights in V^* is within a factor of 2 of the sum of the edge-weights in X. Since both the fractional assignments $\hat{w}$ and w^+ have support X, we get the following claim.

Claim 2.26. *We have:*

$$2 \cdot w^+(E) \geq \sum_{v \in V^*} W_v(w^+) \geq w^+(E).$$

$$2 \cdot \hat{w}(E) \geq \sum_{v \in V^*} W_v(\hat{w}) \geq \hat{w}(E).$$

Since we want to compare the sums of the edge-weights under $\hat{w}$ and w^+, by Claim 2.26 it suffices to focus on the sum of the node-weights in V^* instead. Accordingly, we first lower bound the sum $\sum_{v \in V^*} W_v(\hat{w})$ in Claim 2.27. In the proof, we only use the fact that for each level $i > L'$, the weight of a node $v \in B_i \setminus S_i$ remains roughly the same under the fractional assignments $\hat{w}_i$ and w_i (see Claim 2.12).

Claim 2.27. *We have:*

$$\sum_{v \in V^*} W_v(\hat{w}) \geq (1-\varepsilon) \cdot \sum_{v \in V^*} \sum_{i > L' : v \in B_i \setminus S_i} W_v(w_i).$$

Proof. Fix any node $v \in V^*$. By Claim 2.12, we have: $W_v(\hat{w}_i) \geq (1-\varepsilon) \cdot W_v(w_i)$ at each level $i > L'$ where $v \in B_i \setminus S_i$. Summing over all such levels, we get:

$$\sum_{i > L' : v \in B_i \setminus S_i} W_v(\hat{w}_i) \geq (1-\varepsilon) \cdot \sum_{i > L' : v \in B_i \setminus S_i} W_v(w_i) \tag{29}$$

Since $\hat{w} = \sum_{i > L'} \hat{w}_i$, we have:

$$W_v(\hat{w}) \geq \sum_{i > L' : v \in B_i \setminus S_i} W_v(\hat{w}_i).$$

Hence, equation 29 implies that:

$$W_v(\hat{w}) \geq (1-\varepsilon) \cdot \sum_{i > L' : v \in B_i \setminus S_i} W_v(w_i).$$

We now sum the above inequality over all nodes $v \in V^*$. □

It remains to lower bound the right hand side (RHS) in Claim 2.27 by $\sum_{v \in V^*} W_v(w^+)$. Say that a level $i > L'$ is of Type I, II or III for a node $v \in V^*$ if v belongs to $B_i \setminus S_i$, S_i or $T_i \setminus S_i$ respectively. By Definition 2.6, for every node $v \in V^*$, the set of levels $i > L'$ is partitioned into these three types. The sum in the RHS of Claim 2.27 gives the contribution of the type I levels towards $\sum_{v \in V^*} W_v(w^+)$. In Claims 2.29 and 2.30, we respectively show that the type II and type III levels make negligible contributions towards the sum $\sum_{v \in V^*} W_v(w^+)$. Note that the sum of these contributions from the type I, type II and type III levels *exactly* equals $\sum_{v \in V^*} W_v(w^+)$. Specifically, we have:

$$\sum_{v \in V^*} \sum_{i > L' : v \in B_i \setminus S_i} W_v(w_i) + \sum_{v \in V^*} \sum_{i > L' : v \in S_i} W_v(w_i) + \sum_{v \in V^*} \sum_{i > L' : v \in T_i \setminus S_i} W_v(w_i) = \sum_{v \in V^*} W_v(w^+) \tag{30}$$

Hence, equation 30, Claim 2.29 and Claim 2.30 lead to the following lower bound on the right hand side of Claim 2.27.

Corollary 2.28. *We have:*

$$\sum_{v \in V^*} \sum_{i > L' : v \in B_i \setminus S_i} W_v(w_i) \geq (1 - 40\varepsilon) \cdot \sum_{v \in V^*} W_v(w^+).$$

From Claim 2.27, Corollary 2.28 and equation 3, we get:

$$\sum_{v \in V^*} W_v(\hat{w}) \geq (1/4) \cdot \sum_{v \in V^*} W_v(w^+) \tag{31}$$

Finally, from Claim 2.26 and equation 31, we infer that:

$$\hat{w}(E) \geq (1/2) \cdot \sum_{v \in V^*} W_v(\hat{w}) \geq (1/8) \cdot \sum_{v \in V^*} W_v(w^+) \geq (1/8) \cdot w^+(E)$$

This concludes the proof of Claim 2.17. Accordingly, we devote the rest of this section to the proofs of Claims 2.29 and 2.30.

Claim 2.29. *We have:*

$$\sum_{v \in V^*} \sum_{i > L' : v \in S_i} W_v(w_i) \leq 8\varepsilon \cdot \sum_{v \in V^*} W_v(w^+).$$

Proof. The proof of this claim is very similar to the proof of Claim 2.23 and Corollary 2.24. Going through that proof, one can verify the following upper bound on the number of spurious nodes across all levels $i > L'$.

$$\sum_{i > L'} |S_i| \leq (2\delta L/\varepsilon) \cdot \sum_{v \in V} W_v(w^+) \tag{32}$$

Since each w_i is a fractional matching (see Section 2.1), we have $W_v(w_i) \leq 1$ for all nodes $v \in V$ and all levels $i > L'$. Hence, we get:

$$\sum_{v \in V^*} \sum_{i > L' : v \in S_i} W_v(w_i) \leq \sum_{i > L'} |S_i| \tag{33}$$

From equations 32 and 33, we infer that:

$$\sum_{v \in V^*} \sum_{i > L' : v \in S_i} W_v(w_i) \leq (2\delta L/\varepsilon) \cdot \sum_{v \in V} W_v(w^+) \tag{34}$$

Since the sum of the node-weights under any fractional assignment is equal to twice the sum of the edge-weights, Claim 2.26 implies that:

$$\sum_{v \in V} W_v(w^+) = 2 \cdot w^+(E) \leq 4 \cdot \sum_{v \in V^*} W_v(w^+) \tag{35}$$

Claim 2.29 follows from equations 3, 34 and 35. □

Claim 2.30. *We have:*

$$\sum_{v \in V^*} \sum_{i > L' : v \in T_i \setminus S_i} W_v(w_i) \leq 32\varepsilon \cdot \sum_{v \in V^*} W_v(w^+).$$

Proof. Fix any node $v \in V^*$. By equation 5, we have $\deg_v(E_i) \leq (2\varepsilon d_i/L)$ at each level $i > L'$ where $v \in T_i \setminus S_i$. Further, the fractional matching w_i assigns a weight $1/d_i$ to every edge in its support E_i (see Section 2.1). Combining these two observations, we get: $W_v(w_i) = (1/d_i) \cdot \deg_v(E_i) \leq 2\varepsilon/L$ at each level $i > L'$ where $v \in T_i \setminus S_i$. Summing over all such levels, we get:

$$\sum_{i > L' : v \in T_i \setminus S_i} W_v(w_i) \leq 2\varepsilon \tag{36}$$

If we sum equation 36 over all $v \in V^*$, then we get:

$$\sum_{v \in V^*} \sum_{i > L' : v \in T_i \setminus S_i} W_v(w_i) \leq 2\varepsilon \cdot |V^*| \tag{37}$$

A node $v \in V^*$ has level $\ell(v) > L'$. Hence, all the edges incident upon this node also have level at least $L'+1$. This implies that such a node v receives zero weight from the fractional assignment $w^- = \sum_{i \le L'} w_i$, for any edge in the support of w^- is at level at most L'. Thus, we have: $W_v(w) = W_v(w^+) + W_v(w^-) = W_v(w^+)$ for such a node v. Now, applying Theorem 2.1, we get:

$$1/(1+\varepsilon)^2 \le W_v(w^+) \text{ for all nodes } v \in V^*. \tag{38}$$

Summing equation 38 over all nodes $v \in V^*$ and multiplying both sides by $(1+\varepsilon)^2$, we get:

$$|V^*| \le (1+\varepsilon)^2 \cdot \sum_{v \in V^*} W_v(w^+) \tag{39}$$

Since $(1+\varepsilon)^2 \le 4$ and $V^* \subseteq V$, equations 37, 39 imply that:

$$\sum_{v \in V^*} \sum_{i > L' : v \in T_i \setminus S_i} W_v(w_i) \le 8\varepsilon \cdot \sum_{v \in V} W_v(w^+) \tag{40}$$

The claim follows from equations 35 and 40. □

3 Bipartite graphs

Ideally, we would like to present a dynamic algorithm on bipartite graphs that proves Theorem 1.2. Due to space constraints, however, we will only prove a weaker result stated in Theorem 3.1 and defer the complete proof of Theorem 1.2 to the full version. Throughout this section, we will use the notations and concepts introduced in Section 1.1.

Theorem 3.1. *There is a randomized dynamic algorithm that maintains a* 1.976 *approximation to the maximum matching in a bipartite graph in* $O(\sqrt{n}\log n)$ *expected update time.*

In Section 3.1, we present a result from [5] which shows how to maintain a $(2+\varepsilon)$-approximation to the maximum matching in bipartite graphs in $O(\sqrt{n}/\varepsilon^2)$ update time. In Section 3.2, we build upon this result and prove Theorem 3.1. In Section 3.3, we allude to the extensions that lead us to the proof of Theorem 1.2 in the full version of the paper.

3.1 $(2+\varepsilon)$-approximation in $O(\sqrt{n}/\varepsilon^2)$ update time

The first step is to define the concept of a *kernel*. Setting $\varepsilon = 0, d = 1$ in Definition 3.2, we note that the kernel edges in a $(0,1)$-kernel forms a maximal matching – a matching where every unmatched edge has at least one matched endpoint. For general d, we note that the kernel edges in a $(0,d)$-kernel forms a maximal d-matching – which is a maximal subset of edges where each node has degree at most d. In Lemma 3.3 and Corollary 3.4, we show that the kernel edges in any (ε,d)-kernel preserves the size of the maximum matching within a factor of $2/(1-\varepsilon)$. Since d is the maximum degree of a node in an (ε,d)-kernel, a $(1+\varepsilon)$-approximation to the maximum matching within a kernel can be maintained in $O(d/\varepsilon^2)$ update time using Theorem 1.4. Lemma 3.5 shows that the set of kernel edges themselves can be maintained in $O(n/(\varepsilon d))$ update time. Setting $d = \sqrt{n}$ and combining all these observations, we get our main result in Corollary 3.6.

Definition 3.2. *Fix any* $\varepsilon \in (0,1)$, $d \in [1,n]$. *Consider any subset of nodes* $T \subset V$ *in the graph* $G = (V,E)$, *and any subset of edges* $H \subseteq E$. *The pair* (T,H) *is called an* (ε,d)*-kernel of* G *iff: (1)* $deg_v(H) \le d$ *for all nodes* $v \in V$, *(2)* $deg_v(H) \ge (1-\varepsilon)d$ *for all nodes* $v \in T$, *and (3) every edge* $(u,v) \in E$ *with both endpoints* $u,v \in V \setminus T$ *is part of the subset* H. *We define the set of nodes* $T^c = V \setminus T$, *and say that the nodes in* T *(resp.* T^c*) are "tight" (resp. "non-tight"). The edges in* H *are called "kernel edges".*

Lemma 3.3. *Consider any integral matching $M \subseteq E$ and let (T,H) be any (ε,d)-kernel of $G=(V,E)$ as per Definition 3.2. Then there is a fractional matching w'' in G with support H such that $\sum_{v\in V} W_v(w'') \geq (1-\varepsilon)\cdot|M|$.*

The proof of Lemma 3.3 appears in Section 3.1.1.

Corollary 3.4. *Consider any (ε,d)-kernel as per Definition 3.2. We have $Opt(H) \geq (1/2)\cdot(1-\varepsilon)\cdot Opt(E)$.*

Proof. Let $M \subseteq E$ be a maximum cardinality matching in $G=(V,E)$. Let w'' be a fractional matching with support H as per Lemma 3.3. Since in a bipartite graph the size of the maximum cardinality matching is the same as the size of the maximum fractional matching (see Theorem 1.3), we get: $\text{Opt}(H) = \text{Opt}_f(H) \geq w''(H) = (1/2)\cdot\sum_{v\in V} W_v(w'') \geq (1/2)\cdot(1-\varepsilon)\cdot|M| = (1/2)\cdot(1-\varepsilon)\cdot\text{Opt}(E)$. □

Lemma 3.5. *In the dynamic setting, an (ε,d)-kernel can be maintained in $O(n/(\varepsilon d))$ amortized update time.*

Proof. (Sketch) When an edge (u,v) is inserted into the graph, we simply check if both its endpoints are non-tight. If yes, then we insert (u,v) into H. Next, for each endpoint $x \in \{u,v\}$, we check if $\deg_x(H)$ has now become equal to d due to this edge insertion. If yes, then we delete the node x from T^c and insert it into T. All these operations can be performed in constant time.

Now, consider the deletion of an edge (u,v). If both u,v are non-tight, then we have nothing else to do. Otherwise, for each tight endpoint $x \in \{u,v\}$, we check if $\deg_x(H)$ has now fallen below the threshold $(1-\varepsilon)d$ due to this edge deletion. If yes, then we might need to change the status of the node x from tight to non-tight. Accordingly, we scan through all the edges in E that are incident upon x, and try to insert as many of them into H as possible. This step takes $\Theta(n)$ time in the worst case since the degree of the node x can be $\Theta(n)$. However, the algorithm ensures that this event occurs only after εd edges incident upon x are deleted from E. This is true since we have a *slack* of εd between the largest and smallest possible degrees of a tight node. Thus, we get an amortized update time of $O(n/(\varepsilon d))$. □

Corollary 3.6. *In a bipartite graph, one can maintain a $(2+6\varepsilon)$-approximation to the size of the maximum matching in $O(\sqrt{n}/\varepsilon^2)$ amortized update time.*

Proof. (Sketch) We set $d=\sqrt{n}$ and maintain an (ε,d)-kernel (T,H) as per Lemma 3.5. This takes $O(\sqrt{n}/\varepsilon)$ update time. Next, we note that the maximum degree of a node in H is $d=\sqrt{n}$ (see Definition 3.2). Accordingly, we can apply Theorem 1.4 to maintain a $(1+\varepsilon)$-approximate maximum matching $M_H \subseteq H$ in $O(\sqrt{n}/\varepsilon^2)$ update time. Hence, by Corollary 3.4, this matching M_H is a $2(1+\varepsilon)/(1-\varepsilon) \leq (2+6\varepsilon)$-approximation to the maximum matching in $G=(V,E)$. □

3.1.1 Proof of Lemma 3.3

First, define a fractional assignment w as follows. For every edge $(u,v) \in H$ incident on a tight node, we set $w(e)=1/d$, and for every other edge $(u,v) \in E$, set $w(u,v)=0$. Since each node $v \in V$ has $\deg_v(H) \leq d$, it is easy to check that $W_v(w) \leq 1$ for all nodes $v \in V$. In other words, w forms a fractional matching in G.

Next, we define another fractional assignment w'. First, for every node $v \in T^c$, we define a "capacity" $b(v) = 1 - W_v(w) \in [0,1]$. Next, for every edge $(u,v) \in H \cap M$ whose both endpoints are non-tight, set $w'(u,v) = \min(b(u),b(v))$. For every other edge $(u,v) \in E$, set $w'(u,v)=0$.

We finally define $w''=w+w'$. Clearly, the fractional assignment w'' has support H, since for every edge $(u,v) \in E \setminus H$, we have $w(u,v)=w'(u,v)=0$. Hence, the lemma follows from Claims 3.7 and 3.8.

Claim 3.7. *We have $W_v(w'') \leq 1$ for all nodes $v \in V$, that is, w'' is a fractional matching in G.*

Proof. If a node v is tight, that is, $v \in T$, then we have $W_v(w'') = W_v(w) + W_v(w') = W_v(w) \leq 1$. Hence, for the rest of the proof, consider any node from the remaining subset $v \in T^c = V \setminus T$. There are two cases to consider here.

Case 1. If v is not matched in M, then we have $W_v(w') = 0$, and hence $W_v(w'') = W_v(w) + W_v(w') = W_v(w) \leq 1$.

Case 2. If v is matched in M, then let u be its mate, i.e., $(u,v) \in M$. Here, we have $W_v(w') = w'(u,v) = \min(1 - W_u(w), 1 - W_v(w)) \leq 1 - W_v(w)$. This implies that $W_v(w'') = W_v(w) + W_v(w') \leq 1$. This concludes the proof. □

Claim 3.8. *We have* $\sum_{v \in V} W_v(w'') \geq (1-\varepsilon) \cdot |M|$.

Proof. Throughout the proof, fix any edge $(u,v) \in M$. We will show that $W_u(w'') + W_v(w'') \geq (1-\varepsilon)$. The claim will then follow if we sum over all the edges in M.

Case 1. The edge (u,v) has at least one tight endpoint. Let $u \in T$. In this case, we have $W_u(w'') + W_v(w'') \geq W_u(w'') = W_u(w) + W_u(w') \geq W_u(w) = (1/d) \cdot \deg_u(H) \geq (1-\varepsilon)$.

Case 2. Both the endpoints of (u,v) are non-tight. Without any loss of generality, let $W_u(w) \geq W_v(w)$. In this case, we have $W_u(w'') + W_v(w'') \geq W_u(w'') = W_u(w) + W_u(w') = W_u(w) + w'(u,v) = W_u(w) + \min(1 - W_u(w), 1 - W_v(w)) = W_u(w) + (1 - W_u(w)) = 1$. This concludes the proof. □

3.2 Better than 2-approximation

The approximation guarantee derived in Section 3.1 follows from Claim 3.8. Looking back at the proof of this claim, we observe that we actually proved a stronger statement: Any matching $M \subseteq E$ satisfies the property that $W_u(w'') + W_v(w'') \geq (1-\varepsilon)$ for all matched edges $(u,v) \in M$, where w'' is a fractional matching with support H that depends on M. In the right hand side of this inequality, if we replace the term $(1-\varepsilon)$ by anything larger than 1, then we will get a better than 2 approximation (see the proof of Corollary 3.4). The reason it was not possible to do so in Section 3.1 is as follows. Consider a matched edge $(u,v) \in M$ with $u \in T$ and $v \in T^c$. Since u is tight, we have $1 - \varepsilon \leq W_u(w) = W_v(w'') \leq 1$. Suppose that $W_u(w'') = 1 - \varepsilon$. In contrast, it might well be the case that $W_v(w)$ is very close to being zero (which will happen if $\deg_v(H)$ is very small). Let $W_v(w) \leq \varepsilon$. Also note that $W_v(w'') = W_v(w) + W_v(w') = W_v(w) \leq \varepsilon$ since no edge that gets nonzero weight under w' can be incident on v (for v is already incident upon an edge in M whose other endpoint is tight). Hence, in this instance we will have $W_u(w'') + W_v(w'') \leq (1-\varepsilon) + \varepsilon = 1$, where $(u,v) \in M$ is a matched edge with one tight and one non-tight endpoint.

The above discussion suggests that we ought to "throw in" some additional edges into our kernel – edges whose one endpoint is tight and the other endpoint is non-tight with a very small degree in H. Accordingly, we introduce the notion of *residual edges* in Section 3.2.1. We show that the union of the kernel edges and the residual edges preserves the size of the maximum matching within a factor of strictly less than 2. Throughout the rest of this section, we set the values of two parameters δ, ε as follows.

$$\delta = 1/20, \varepsilon = 1/2000 \tag{41}$$

3.2.1 The main framework: Residual edges

We maintain an (ε, d)-skeleton (T, H) as in Section 3.1. We further partition the set of non-tight nodes $T^c = V \setminus T$ into two subsets: $B \subseteq T^c$ and $S = T^c \setminus B$. The set of nodes in B (resp. S) are called "big" (resp. "small"). They satisfy the following degree-thresholds: (1) $\deg_v(H) \leq 2\delta d/(1-\delta)$ for all small nodes $v \in S$, and (2) $\deg_v(H) \geq (2\delta - \varepsilon)d/(1-\delta)$ for all big nodes $v \in B$. Let $E^r = \{(u,v) \in E : u \in T, v \in S\}$ be the subset of edges joining the tight and the small nodes. We maintain a *maximal* subset of edges $M^r \subseteq E^r$ subject to the following constraints: (1) $\deg_v(M^r) \leq 1$ for all tight nodes $v \in T$ and (2) $\deg_v(M^r) \leq 2$ for all

small nodes $v \in S$. The edges in M^r are called the "residual edges". The degree of a node in M^r is called its "residual degree". The corollary below follows from the maximality of the set $M^r \subseteq E^r$.

Corollary 3.9. *If an edge $(u,v) \in E^r$ with $u \in T$, $v \in S$ is not in M^r, then either $deg_v(M^r) = 1$ or $deg_u(M^r) = 2$.*

Lemma 3.10. *We can maintain the set of kernel edges H and the residual edges M^r in $O(n \log n/(\varepsilon d))$ update time.*

Proof. (Sketch) We maintain an (ε, d)-kernel as per the proof of Lemma 3.5. We maintain the node-sets $B, S \subseteq T^c$ and the edge-set E^r in the same lazy manner: A node changes its status only after $\Omega(\varepsilon d)$ edges incident upon it are either inserted into or deleted from G (since δ is a constant), and when that happens we might need to make $\Theta(n)$ edge insertions/deletions in E^r. This gives the same amortized update time of $O(n/(\varepsilon d))$ for maintaining the edge-set E^r.

In order to maintain the set of residual edges $M^r \subseteq E^r$, we use a simple modification of the dynamic algorithm of Baswana et al. [2] that maintains a maximal matching in $O(\log n)$ update time. This holds since M^r is a *maximal b-matching* in E^r where each small node can have at most two matched edges incident upon it, and each tight node can have at most one matched edge incident upon it. □

Lemma 3.11. *Fix any (ε, d) kernel (T, H) as in Section 3.1, any set of residual edges M^r as in Section 3.2.1, and any matching $M \subseteq E$. Then we have a fractional matching w'' on support $H \cup M^r$ such that $\sum_{v \in V} W_v(w'') \geq (1 + \delta/4) \cdot |M|$.*

Roadmap for the rest of this section. The statement of Lemma 3.11 above is similar to that of Lemma 3.3 in Section 3.1. Hence, using a similar argument as in Corollary 3.4, we infer that the set of edges $M^r \cup H$ preserves the size of the maximum matching within a factor of $2/(1+\delta/4)$. Since $\deg_v(M^r \cup H) = \deg_v(H) + \deg_v(M^r) \leq d + 2$ for all nodes $v \in V$ (see Definition 3.2), we can maintain a $(1+\varepsilon)$-approximate maximum matching in $H \cup M^r$ using Theorem 1.4 in $O(d/\varepsilon^2)$ update time. This matching will give a $2(1+\varepsilon)/(1+\delta/4) = 1.976$-approximation to the size of maximum matching in G (see equation 41). The total update time is $O(d/\varepsilon^2 + n \log n/(\varepsilon d))$, which becomes $O(\sqrt{n} \log n)$ if we set $d = \sqrt{n}$ and plug in the value of ε. This concludes the proof of Theorem 3.1.

It remains to prove Lemma 3.11, which is done in Section 3.2.2.

3.2.2 Proof of Lemma 3.11

We will define four fractional assignments w, w^r, w', w''. It might be instructive to contrast the definitions of the fractional assignments w, w' and w'' here with Section 3.1.1.

The fractional assignment w: Set $w(e) = (1-\delta)/d$ for every edge $e \in H$ incident upon a tight node. Set $w(e) = 0$ for every other edge $e \in E$. Hence, we have $W_v(w) = ((1-\delta)/d) \cdot \deg_v(H)$ for all nodes $v \in V$. Accordingly, recalling the bounds on $\deg_v(H)$ for various types of nodes (see Definition 3.2, Section 3.2.1), we get:

$$W_v(w) \leq (1-\delta) \text{ for all nodes } v \in V. \tag{42}$$

$$W_v(w) \leq 2\delta \text{ for all small nodes } v \in S. \tag{43}$$

$$W_v(w) \geq (1-\delta)(1-\varepsilon) \text{ for all tight nodes } v \in T. \tag{44}$$

$$W_v(w) \geq 2\delta - \varepsilon \text{ for all big nodes } v \in B. \tag{45}$$

The fractional assignment w^r: Set $w^r(e) = \delta$ for every residual edge $e \in M^r$. Set $w^r(e) = 0$ for every other edge $e \in E$.

The fractional assignment w': For every node $v \in T^c$, we define a "capacity" $b(v)$ as follows. If $v \in B \subseteq T^c$, then $b(v) = 1 - W_v(w)$. Else if $v \in S = T^c \setminus B$, then $b(v) = 1 - 2\delta - W_v(w)$. Hence, equations 42, 43 imply that:

$$b(v) \geq \delta \text{ for all big nodes } v \in B. \tag{46}$$

$$b(v) \geq 1 - 4\delta \text{ for all small nodes } v \in S. \tag{47}$$

For every edge $(u,v) \in M$ with $u,v \in T^c = V \setminus T$, we set $w'(u,v) = \min(b(u), b(v))$. For every other edge $e \in E$, we set $w'(e) = 0$. By Definition 3.2, every edge whose both endpoints are non-tight is a kernel edge. Hence, an edge gets nonzero weight under w' only if it is part of the kernel.

The fractional assignment w'': Define $w'' = w + w^r + w'$.

Roadmap for the rest of the proof. Each of the fractional assignments w, w^r, w' assigns zero weight to every edge $e \in E \setminus (H \cup M^r)$. Hence, the fractional assignment $w'' = w + w^r + w'$ has support $H \cup M^r$. In Claim 3.12, we show that w'' is a fractional matching in G. Moving on, in Definition 3.13, we partition the set of matched edges in M into two parts. The subset $M_1 \subseteq M$ consists of those matched edges that have one tight and one small endpoints, and the subset $M_2 = M \setminus M_1$ consists of the remaining edges. In Claims 3.14 and 3.15, we relate the node-weights under w, w', w^r with the sizes of the matchings M_1 and M_2. Adding up the bounds from Claims 3.14 and 3.15, Corollary 3.16 lower bounds the sum of the node-weights under w'' by the size of the matching M. Finally, Lemma 3.11 follows from Claim 3.12 and Corollary 3.16.

Claim 3.12. *We have $W_v(w'') \leq 1$ for all nodes $v \in V$.*

Proof. Consider any node $v \in V$. By equation 42, we have: $W_v(w) \leq 1 - \delta$. Also note that $W_v(w^r) \leq \delta$ for all tight nodes $v \in T$, $W_v(w^r) \leq 2\delta$ for all small nodes $v \in S$, and $W_v(w^r) = 0$ for all big nodes $v \in B$. This holds since the degree (among the edges in M^r) of a tight, small and big node is at most one, two and zero respectively. Next, note that for all nodes $v \in T^c$, we have $W_v(w') \leq b(v)$. This holds since there is at most one edge in $M \cap H$ incident upon v (since M is a matching). So at most one edge incident upon v gets a nonzero weight under w', and the weight of this edge is at most $b(v)$. Finally, note that every edge with nonzero weight under w' has both the endpoints in T^c. Hence, we have $W_v(w') = 0$ for all tight nodes $v \in T$. To complete the proof, we now consider three possible cases.

Case 1. $v \in T$. Here, $W_v(w'') = W_v(w) + W_v(w^r) + W_v(w') = W_v(w) + W_v(w^r) \leq W_v(w) + \delta \leq (1-\delta) + \delta = 1$.

Case 2. $v \in S$. Here, $W_v(w'') = W_v(w) + W_v(w^r) + W_v(w') \leq W_v(w) + 2\delta + b(v) = W_v(w) + 2\delta + (1 - 2\delta - W_v(w)) = 1$.

Case 3. $v \in B$. Here, $W_v(w'') = W_v(w) + W_v(w^r) + W_v(w') = W_v(w) + W_v(w') \leq W_v(w) + b(v) = W_v(w) + (1 - W_v(w)) = 1$. □

Definition 3.13. *Partition the set of edges in M into two parts: $M_1 = \{(u,v) \in M : u \in T, v \in S\}$ and $M_2 = M \setminus M_1$.*

Claim 3.14. *Recall Definition 3.13. We have:*

$$\sum_{(u,v) \in M_2} W_u(w + w') + W_v(w + w') \geq (1 + \delta/4) \cdot |M_2|.$$

Proof. Fix any edge $(u,v) \in M_2$, and let LHS $= W_u(w + w') + W_v(w + w')$. We will show that LHS $\geq (1 + \delta/4)$. The claim will then follow if we sum over all such edges M_2. We recall equation 41 and consider four possible cases.

Case 1. Both endpoints are tight, that is, $u,v \in T$. Here, from equation 44 we get: LHS $\geq 2\cdot(1-\delta-\varepsilon+\delta\varepsilon) \geq (1+\delta/4)$.

Case 2. One endpoint is tight, and one endpoint is big, that is, $u \in T$, $v \in B$. Here, from equations 44, 45 we get: LHS $\geq (1-\delta-\varepsilon+\delta\varepsilon)+(2\delta-\varepsilon) \geq (1+\delta-2\varepsilon) \geq 1+\delta/4$.

Case 3. Both endpoints are non-tight, that is, $u,v \in B \cup S$. Without any loss of generality, let $b(u) \geq b(v)$. Note that $(u,v) \in H$ since both $u,v \in T^c$, and hence $(u,v) \cap M_2 \cap H$. Thus, we have $W_u(w') = W_v(w') = w'(u,v) = b(v)$ since at most one matched edge can be incident upon a node. Now, either $v \in B$ or $v \in S$. In the former case, from equation 47 we get: LHS $\geq W_u(w') + W_v(w') = 2\cdot b(v) \geq 2(1-4\delta) \geq (1+\delta/4)$. In the latter case, from equation 46 we get: LHS $\geq (W_v(w)+W_v(w'))+W_u(w') = (b(v)+W_v(w))+b(v) = 1+b(v) \geq 1+\delta \geq 1+\delta/4$. □

Claim 3.15. *Recall Definition 3.13. We have:*

$$\sum_{(u,v)\in M_1} (W_u(w)+W_v(w)) + \sum_{v\in V} W_v(w^r) \geq (1+\delta/4)\cdot|M_1|.$$

Proof. Every edge $(u,v) \in M_1$ has one endpoint $u \in T$. Thus, Applying equation 44 we get: $W_u(w) + W_v(w) \geq W_u(w) \geq 1-\delta-\varepsilon$. Summing over all such edges, we get:

$$\sum_{(u,v)\in M_1} W_u(w)+W_v(w) \geq (1-\delta-\varepsilon)\cdot|M_1| \tag{48}$$

Recall that $\deg_u(M^r) \leq 1$ for every tight node $u \in T$. Accordingly, we classify each tight node as being either "full" (in which case $\deg_u(M^r)=1$) or "deficient" (in which case $\deg_u(M^r)=0$). Further, recall that each edge $(u,v) \in M_1$ has one tight and one small endpoints. We say that an edge $(x,y) \in M_1$ is *deficient* if the tight endpoint of the edge is deficient. Now, consider any deficient edge $(x,y) \in M_1$ where $x \in T$ and $y \in S$. Since $\deg_x(M^r)=0$, it follows that $(x,y) \in E^r \setminus M^r$. From the maximality of M^r, we infer that $\deg_y(M^r)=2$. Accordingly, there must be two edges $(x',y),(x'',y) \in M^r$ with $x',x'' \in T$. It follows that both the nodes x',x'' are full. We say that the tight nodes x',x'' are *conjugates* of the deficient edge $(x,y) \in M_1$. In other words, we have shown that every deficient edge in M_1 has two conjugate tight nodes. Further, the same tight node x' cannot be a conjugate of two different deficient edges in M_1, for otherwise each of those deficient edges will contribute one towards $\deg_{x'}(M^r)$, and we will get $\deg_{x'}(M^r) \geq 2$, which is a contradiction. Thus, a simple counting argument implies that the number of conjugate tight nodes is exactly twice the number of deficient matched edges in M_1. Let $D(M_1), F, C$ respectively denote the set of deficient matched edges in M_1, the set of full tight nodes and the set of conjugate tight nodes. Thus, we get:

$$T \supseteq F \supseteq C, \;\; D(M_1) \subseteq M_1, \text{ and } |C| = 2\cdot|D(M_1)| \tag{49}$$

Now, either $|D(M_1)| \leq (1/3)\cdot|M_1|$ or $|D(M_1)| > (1/3)\cdot|M_1|$. In the former case, at least a $(2/3)^{rd}$ fraction of the edges in M_1 are not deficient, and each such edge has one tight endpoint that is full. Thus, we get $|F| \geq (2/3)\cdot|M_1|$. In the latter case, from equation 49 we get $|F| \geq |C| = 2\cdot|D(M_1)| > (2/3)\cdot|M_1|$. Thus, in either case we have $|F| \geq (2/3)\cdot|M_1|$. Since each node $v \in F \subseteq T$ has $\deg_v(M^r)=1$, and since each edge $e \in M^r$ has weight $w^r(e)=\delta$, it follows that $W_v(w^r)=\delta$ for all nodes $v \in F \subseteq T$. Hence, we get $\sum_{v\in T} W_v(w^r) \geq \delta\cdot|F| \geq (2\delta/3)\cdot|M_1|$. Next, we note that each edge in M^r contributes the same amount δ towards the weights of both its endpoints – one tight and the other small. Thus, we have:

$$\sum_{v\in S} W_v(w^r) = \sum_{v\in T} W_v(w^r) \geq (2\delta/3)\cdot|M_1|.$$

Since $B \cup S \subseteq V$ and $B \cap S = \emptyset$, we get:

$$\sum_{v\in V} W_v(w^r) \geq \sum_{v\in B\cup S} W_v(w^r) \geq (4\delta/3)\cdot|M_1|.$$

This inequality, along with equation 48, gives us:

$$\begin{aligned}&\sum_{(u,v)\in M_1}(W_u(w)+W_v(w))+\sum_{v\in V}W_v(w^r)\\ \geq (1-\delta-\varepsilon)\cdot|M_1|+(4\delta/3)\cdot|M_1| &= (1+\delta/3-\varepsilon)\cdot|M_1|\\ &\geq (1+\delta/4)\cdot|M_1|.\end{aligned}$$

The last inequality follows from equation 41. □

Corollary 3.16. *We have:*

$$\sum_{v\in V}W_v(w'')\geq(1+\delta/4)\cdot|M|.$$

Proof. Since $|M|=|M_1|+|M_2|$, the corollary follows from adding the inequalities stated in Claims 3.14 and 3.15, and noting that no node-weight under w'' is counted twice in the left hand side. □

3.3 Extensions

We gave a randomized algorithm for maximum bipartite matching that maintains a better than 2 approximation in $O(\sqrt{n}\log n)$ update time. In the full version of the paper, we derandomize this scheme using the following idea. Instead of applying the randomized maximal matching algorithm from [2] for maintaining the set of residual edges M^r, we maintain a *residual fractional matching* using the deterministic algorithm from [5] (see Theorem 2.1). To carry out the approximation guarantee analysis, we have to change the proof of Lemma 3.11 (specifically, the proof of Claim 3.15).

To get arbitrarily small polynomial update time, we maintain a partition of the node-set into multiple levels. The top level consists of all the tight nodes (see Definition 3.2). We next consider the subgraph induced by the non-tight nodes. Each edge in this subgraph is a kernel edge (see Definition 3.2). Intuitively, we split the node-set of this subgraph again into two parts by defining a kernel within this subgraph. The tight nodes we get in this iteration forms the next level in our partition of V. We keep doing this for K levels, where K is a sufficiently large integer. We show that (a) this structure can be maintained in $O(n^{2/K})$ update time, and (b) by combining the fractional matchings from all these levels, we can get an α_K approximate maximum fractional matching in G, where $1\leq\alpha_K<2$. By Theorem 1.3, this gives α_K-approximation to the size of the maximum integral matching in G. See the full version of the paper for the details.

4 Open Problems

In this paper, we presented two deterministic dynamic algorithms for maximum matching. Our first algorithm maintains a $(2+\varepsilon)$-approximate maximum matching in a general graph in $O(\text{poly}(\log n,1/\varepsilon))$ update time. The exponent hidden in the polylogorithmic factor of the update time, however, is rather huge. It will be interesting to bring down the update time of this algorithm to $O(\log n/\varepsilon^2)$ without increasing the approximation factor. This will match the update time in [5] for maintaining a *fractional* matching.

We also showed how to maintain a better than 2 approximation to the size of the maximum matching on bipartite graphs in $O(n^{2/K})$ update time, for every sufficiently large integer K. The approximation ratio approaches 2 as K becomes large. The main open problem here is to design a dynamic algorithm that gives better than 2 approximation in polylogarithmic update time. This remains open even on bipartite graphs and even if one allows randomization.

References

[1] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 18-21, 2014, Philadelphia, PA, USA*, pages 434–443, 2014. 1, 2

[2] Surender Baswana, Manoj Gupta, and Sandeep Sen. Fully dynamic maximal matching in $\mathcal{O}(\log n)$ update time. *SIAM J. Comput.*, 44(1):88–113, 2015. Announced at FOCS 2011. 1, 2, 20, 23

[3] Aaron Bernstein and Cliff Stein. Fully dynamic matching in bipartite graphs. In *ICALP*, pages 167–179, 2015. 1, 2

[4] Aaron Bernstein and Cliff Stein. Faster fully dynamic matchings with small approximation ratios. In *SODA*, 2016. to appear. 1

[5] Sayan Bhattacharya, Monika Henzinger, and Giuseppe F. Italiano. Deterministic fully dynamic data structures for vertex cover and matching. *CoRR*, abs/1412.1318, 2014. 1, 2, 3, 4, 17, 23, 27, 115

[6] Michael Crouch and Daniel S. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, volume 28 of *LIPIcs*, pages 96–104. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014. 2

[7] Manoj Gupta and Richard Peng. Fully dynamic $(1+\varepsilon)$-approximate matchings. In *54th IEEE Symposium on Foundations of Computer Science*, pages 548–557, 2013. 1, 2, 3, 28, 114

[8] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. *STOC*, pages 21–30, 2015. 1, 2

[9] Amos Israeli and Y. Shiloach. An improved parallel algorithm for maximal matching. *Information Processing Letters*, 22:57–60, 1986. 2, 48

[10] Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *APPROX-RANDOM*, volume 7408 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2012. 2

[11] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *SODA*, 2016. to appear. 1, 2

[12] Ofer Neiman and Shay Solomon. Simple deterministic algorithms for fully dynamic maximal matching. In *45th ACM Symposium on Theory of Computing*, pages 745–754, 2013. 1

[13] Krzysztof Onak and Ronitt Rubinfeld. Maintaining a large matching and a small vertex cover. In *42nd ACM Symposium on Theory of Computing*, pages 457–464, 2010. 1

[14] Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In Leonard J. Schulman, editor, *STOC*, pages 603–610. ACM, 2010. 2

[15] David Peleg and Shay Solomon. Dynamic $(1+\delta)$-approximate matchings: A density-sensitive approach. In *SODA*, 2016. to appear. 1, 2

[16] Piotr Sankowski. Faster dynamic matchings and vertex connectivity. In *18th ACM-SIAM Symposium on Discrete Algorithms*, pages 118–126, 2007. 1

[17] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz.*, 3:25 – 30, 1964. 38, 39, 42

[18] V. G. Vizing. The chromatic class of a multigraph. *Kibernetika*, 3:29–39, 1965. 2, 38, 39, 42

Part II

DYNAMIC ALGORITHM FOR GENERAL GRAPHS: FULL DETAILS

5 Preliminaries

In this part of the writeup, we will show how to maintain a $(2+\varepsilon)$-approximate maximum matching in a general graph with $O(\text{poly}(\log n, 1/\varepsilon))$ update time.

Notations.

A "matching" $M \subseteq E$ in a graph $G = (V,E)$ is a subset of edges that do not share any common endpoints. The "size" of a matching is the number of edges contained in it. We will also be interested in the concept of a "fractional matching". Towards this end, consider any weight function $w : E \to [0,1]$ that assigns fractional weights to the edges in the graph. Given any such weight function, we let $W(v,E') = \sum_{(u,v)\in E'} w(u,v)$ denote the total weight a node $v \in V$ receives from its incident edges in $E' \subseteq E$. We also define $w(E') = \sum_{e\in E'} w(e)$ to be the sum of the weights assigned to the edges in $E' \subseteq E$. Now, the weight function w is a "fractional matching" iff we have $W(v,E) \leq 1$ for all $v \in V$. The "value" of this fractional matching w is given by $w(E)$. Given a graph $G = (V,E)$, we will let OPT_G (resp. OPT_G^f) denote the maximum "size of a matching" (resp. "value of a fractional matching") in G. Then it is well known that $\text{OPT}_G \leq \text{OPT}_G^f \leq (3/2)\cdot \text{OPT}_G$. To see an example where the second inequality is tight, consider a triangle-graph and assign a weight $1/2$ to each edge in this triangle. This gives a fractional matching of value $3/2$, whereas every matching in this graph is of size one. We say that a fractional matching w is "λ-maximal", for $\lambda \geq 1$, iff for every edge $(u,v) \in E$, we have $W(u,E) + W(v,E) \geq 1/\lambda$. The value of a λ-maximal fractional matching in G is a 2λ-approximation to OPT_G^f. Throughout the paper, we will use the symbol $\deg(v,E')$ to denote the number of edges in a subset $E' \subseteq E$ that are incident upon a node $v \in V$.

Maintaining a large fractional matching:

The paper [5] maintains a $(2+\varepsilon)$-approximate maximum fractional matching in $O(\log n/\varepsilon^2)$ update time. We will now summarize their result. Consider an input graph $G = (V,E)$ with $|V| = n$ nodes and $|E| = m$ edges. Fix any two constants $\alpha, \beta \geq 1$ and define $L = \lceil \log_\beta(n/\alpha) \rceil$. An "$(\alpha,\beta)$-partition" of the input graph $G = (V,E)$ partitions the node-set V into $(L+2)$ subsets $\{V_i\}, i \in \{-1,0,1,\ldots,L\}$. We say that the nodes belonging to the subset V_i are in "level i". We denote the level of a node v by $\ell(v)$, i.e., $v \in V_i$ iff $\ell(v) = i$. We next define the "level of an edge" (u,v) to be $\ell(u,v) = \max(\ell(u),\ell(v))$. In other words, the level of an edge is the maximum level of its endpoints. We let $E_i = \{e \in E : \ell(e) = i\}$ denote the set of edges at level i, and define the subgraph $G_i = (V,E_i)$. This partitions the edge-set E into $(L+2)$ subsets $\{E_i\}, i \in \{-1,0,1,\ldots,L\}$. Finally, for each level i, we define the value $d_i = \beta^i \cdot (\alpha\beta)$, and assign a weight $w(e) = 1/d_i$ to all the edges $e \in E_i$. We require that an (α,β)-partition satisfy the invariants below.

Invariant 5.1. *For every edge $e \in E$, we have $\ell(e) \geq 0$. In other words, every edge has at least one endpoint v at level $\ell(v) \geq 0$, or, equivalently $E_{-1} = \emptyset$. This means that $V \setminus V_{-1}$ forms a vertex cover in G.*

Invariant 5.2. *We have $W(v,E) \in [1/(\alpha\beta), 1]$ for all nodes $v \in V \setminus V_{-1}$, and $W(v,E) \leq 1$ for all nodes $v \in V_{-1}$. This, along with Invariant 5.1, means that w is an $(\alpha\beta)$-maximal fractional matching in G.*

Corollary 5.3. *In an (α,β)-partition, every node $v \in V$ has $deg(v,E_i) \leq d_i$ for all levels $i \in [0,L]$. Thus, we refer to d_i as being the "degree-threshold" of level i.*

Proof. Invariant 5.1 implies that $W(v,E_i) = \deg(v,E_i)\cdot(1/d_i) \leq 1$ □

Theorem 5.4. *[5] Set $\alpha = 1+3\varepsilon$ and $\beta = 1+\varepsilon$, which means $L = O(\log n/\varepsilon)$. We can maintain the node-sets $\{V_i\}$, the edge-sets $\{E_i\}$ and the edge-weights $\{w(e)\}$ in an (α,β)-partition of G in $O(\log n/\varepsilon^2)$ update time. Hence, the edge-weights form a $(2\alpha\beta)$-approximate maximum fractional matching in G.*

Maintaining a near-optimal matching in a bounded degree graph:

Gupta and Peng [7] maintains a $(1+\varepsilon)$-approximate maximum matching in $O(\sqrt{m}/\varepsilon^2)$ time. The next theorem follows as an immediate consequence of their algorithm.

Theorem 5.5. *If we are guaranteed that the maximum degree in a graph $G=(V,E)$ never exceeds some threshold d, then we can maintain a $(1+\varepsilon)$-approximate maximum matching in G in $O(d/\varepsilon^2)$ update time.*

Problem definition.

The input graph $G=(V,E)$ is changing via a sequence of "edge-updates", where the term "edge-update" refers to the insertion/deletion of an edge in E. The node-set V remains fixed. In the beginning, we have an empty graph, i.e., $E=\emptyset$. We will show how to maintain a $(2+\varepsilon)$-approximate maximum matching in G in $O(\text{poly}(\log n, 1/\varepsilon))$ amortized update time, where $n=|V|$ is the number of nodes in the input graph.

5.1 Setting some parameter values

Throughout this paper, we will be dealing with the parameters defined below. We let $n=|V|$ denote the number of nodes in the input graph. Basically, we treat ε and γ as small positive constants, and $\delta = \Theta(1/\text{poly}\log n)$. It should be clear to the reader that as long as the number of nodes n is sufficiently large, all the equations stated below will be satisfied. We will use these equations throughout the rest of the paper.

$$0<\varepsilon<1 \tag{50}$$
$$\alpha = 1+3\varepsilon \tag{51}$$
$$\beta = 1+\varepsilon \tag{52}$$
$$L = \lceil \log_\beta(n/\alpha) \rceil \tag{53}$$
$$L \text{ is a positive integer and } L = O(\log n/\varepsilon) \text{ and } \lfloor L^4/2 \rfloor \geq L^4/4 \geq (1/\varepsilon). \tag{54}$$
$$0<\delta<1 \tag{55}$$
$$d \text{ is a positive integer and } L^4 \leq d \leq n. \tag{56}$$
$$L' = \left\lceil \log_\beta \left(\frac{2L^4}{\alpha\beta} \right) \right\rceil \tag{57}$$
$$L_d = \left\lceil \log_2 \left(\frac{d}{L^4} \right) \right\rceil \tag{58}$$
$$2^{L_d} = \frac{d}{\lambda_d L^4} \text{ where } 1/2 \leq \lambda_d \leq 1. \tag{59}$$
$$L = O(\log n/\varepsilon) \text{ and } L_d \leq L \tag{60}$$
$$\delta < 1/2 \tag{61}$$

$$L \geq L_{d_i} \text{ for all } i \in [L', L] \tag{62}$$
$$L^2 \geq 4/\varepsilon \tag{63}$$
$$L \geq 3\alpha\beta \tag{64}$$
$$L \geq 3 \tag{65}$$
$$L_d \geq \gamma \tag{66}$$
$$L \geq 4L_d/\gamma \tag{67}$$
$$L^2 \geq (8e^\gamma L_d)/(\varepsilon\gamma\lambda_d) \tag{68}$$

We next define the parameters h_0, h_1, h_2. Note that for all $k \in [0,2]$, we have $h_k \leq 1$, and, furthermore, h_k can be made very close to one if the parameters $\delta, \varepsilon, \gamma$ are sufficiently small and K is sufficiently large.

$$h_0 = e^{-\gamma}\cdot(1+4\varepsilon)^{-1} \quad (69)$$

$$1/2 \le h_0 \le (1+4\varepsilon)^{-1} \quad (70)$$

$$h_1 = h_0\cdot e^{-\gamma}\cdot((\alpha\beta)^{-1}-3\varepsilon) \quad (71)$$

$$h_2 = (1-32\delta L^4/\varepsilon^2)\cdot(h_1-\varepsilon-1/K) \text{ where } K \text{ is a large positive integer.} \quad (72)$$

5.2 Skeleton of a graph

Set the parameter values as in Section 5.1. Throughout the paper, we consider an (α,β)-partition (see Section 5) of the graph $G=(V,E)$. Below, we introduce the concept of a skeleton of a graph.

Definition 5.6. *Consider a graph $G=(V,E)$ with n nodes and maximum degree d, that is, $deg(v,E)\le d$ for all $v\in V$. A tuple (B,T,S,X) with $B,T,S\subseteq V$ and $X\subseteq E$, is a "skeleton" of G if and only if:*

1. *We have $B\cup T=V$ and $B\cap T=\emptyset$. In other words, the node-set V is partitioned into subsets B and T. The nodes in B are called "big", whereas the nodes in T are called "tiny".*
2. *For every big node $v\in B$, we have $deg(v,E)>\varepsilon d/L^2$, where L is the same as in the (α,β) partition.*
3. *For every tiny node $v\in T$, we have $deg(v,E)<3\varepsilon d/L^2$.*
4. *We have $|S|\le 4\delta\cdot|B|$. The nodes in S are called "spurious".*
5. *Recall equations 56, 58 and 59. For every big, non-spurious node $v\in B\setminus S$, we have:*

$$e^{-\gamma}\cdot(\lambda_d L^4/d)\cdot deg(v,E)\le deg(v,X)\le e^{\gamma}\cdot(\lambda_d L^4/d)\cdot deg(v,E).$$

6. *For every node $v\in\mathcal{V}$, we have $deg(v,X)\le\lambda_d L^4+2$.*
7. *For every tiny, non-spurious node $v\in T\setminus S$, we have $deg(v,X)\le 3\varepsilon\lambda_d L^2+2$.*

Thus, in a skeleton (B,T,S,X) of a graph $G=(V,E)$, each node is classified as being either "big" or "tiny", but not both. Barring a few exceptions (which are called "spurious" nodes), the big nodes have large degrees and the tiny nodes have small degrees. The number of spurious nodes, however, is negligible in comparison with the number of big nodes. In a skeleton, the degrees of the big, non-spurious nodes are scaled down by a factor very close to $(\lambda_d L^4/d)$, where d is the maximum degree of a node in G. Further, the maximum degree of any node in a skeleton is $\lambda_d L^4+2$. Finally, the maximum degree of any tiny, non-spurious node in the skeleton is $3\varepsilon\lambda_d L^2+2$. The next theorem shows that we can efficiently maintain the edge-set of a skeleton of a graph.

Theorem 5.7. *Given a graph $G=(V,E)$ with n nodes, we can maintain the edges $e\in H$ of a skeleton (B,T,S,H) of G under a sequence of edge insertions/deletions in E. We assume that $E=\emptyset$ in the beginning. Our algorithm is deterministic and has an amortized update time of $O(L^2L_d^3/(\varepsilon\gamma\delta))$. See equation 58.*

The next theorem will be helpful in deriving the approximation guarantee of our algorithm.

Theorem 5.8. *Consider an (α,β)-partition of the graph $G=(V,E)$. For every level $i\in[L',L]$, let (B_i,T_i,S_i,X_i) be a skeleton of $G_i=(V,E_i)$. Define the edge-sets $X=\bigcup_{i=L'}^{L}X_i$, and $Y=\bigcup_{i=0}^{L'-1}E_i$. Then the following conditions are satisfied.*

1. *For every node $v\in V$, we have $deg(v,X\cup Y)=O(L^5)$.*
2. *The size of the maximum cardinality matching in $G(X\cup Y)=(V,X\cup Y)$ is a $(2/h_2)\cdot(1+\varepsilon)$-approximation to the size of the maximum cardinality matching in $G=(V,E)$. See equation 72.*

Our main result is summarized in Theorem 5.9, which follows from Theorems 5.7 and 5.8.

Theorem 5.9. *In a dynamic setting, we can deterministically maintain a $(2+\varepsilon)$-approximate maximum matching in a graph $G=(V,E)$ in $O(poly\,(\log n,\varepsilon^{-1}))$ amortized update time.*

Proof. We set the parameters γ and δ as follows.

$$\gamma = \varepsilon, \delta = \varepsilon^3/L^4.$$

The input graph $G = (V,E)$ changes via a sequence of edge-updates in E. In this setting, we maintain the edge-sets $\{E_i\}$ of an (α,β)-partition of G as per Theorem 5.4. This requires $O(\log n/\varepsilon^2)$ update time. Next, for each level $i \in [L',L]$, we maintain the edge-set $X_i \subseteq E_i$ as per Theorem 5.7. By equation 60, this requires $O(\text{poly}\,(\log n, (\varepsilon\gamma\delta)^{-1})) = O(\text{poly}\,(\log n, \varepsilon^{-1}))$ update time. Finally, by Theorem 5.8, the maximum degree of a node in the subgraph $G(X \cup Y)$ is $O(L^4)$. Hence, we maintain a $(1+\varepsilon)$-approximate maximum matching $M' \subseteq (X \cup Y)$ in the subgraph $G(X \cup Y)$ as per Theorem 5.5. This also requires $O(\text{poly}(\log n, \varepsilon^{-1}))$ time. Accordingly, the total update time of the algorithm is $O(\text{poly}(\log n, \varepsilon^{-1}))$.

By Theorem 5.8, the maximum matching in $G(X \cup Y)$ is a $(2/h_2)\cdot(1+\varepsilon)$-approximation to the maximum matching in G. Recall that our algorithm maintains M', which is a $(1+\varepsilon)$-approximation to the maximum matching in $G(X \cup Y)$. Thus, M' is a $(2/h_2)\cdot(1+\varepsilon)^2$-approximation to the maximum matching in G. Since $\delta = \varepsilon^3/L^4$ and $\gamma = \varepsilon$, equations 69, 71 and 72 imply that $(1/h_2)\cdot(1+\varepsilon)^2 = 1+O(\varepsilon)$ for sufficiently small ε.

To summarize, we maintain a matching $M' \subseteq X \cup Y \subseteq E$. The size of this matching M' is a $(2+O(\varepsilon))$-approximation to the size of the maximum cardinality matching in G. Further, our algorithm for maintaining M' requires $O(\text{poly}\,(\log n, \varepsilon^{-1}))$ update time. This concludes the proof of Theorem 5.9. □

In Section 6 we prove Theorem 6, and in Section 7 we prove Theorem 5.7.

6 Deriving the approximation guarantee: Proof of Theorem 5.8

The first part of the theorem, which upper bounds the maximum degree a node can have in $X \cup Y$, is relatively easy to prove. The basic idea is that for any given level $i \in [0,L]$, the degree of a node is $O(L^4)$ among the level-i edges in $X \cup Y$. This is true for the levels $i \geq L'$ because of the condition (6) in Definition 5.6. On the other hand, for $i < L'$, every edge $e \in E_i$ has weight $w(e) = \Omega(1/L^4)$. Thus, the degree of a node cannot be too high among those edges, for otherwise the total weight received by the node would exceed one. This is summarized in the lemma below.

Lemma 6.1. *We have $\deg(v, X \cup Y) = O(L^5)$ for all nodes $v \in V$.*

Proof. Consider any node $v \in V$. By Definition 5.6, we have $\deg(v, X_i) = O(L^4)$ for all levels $i \in [L',L]$. Next, note that at every level $i < L'$, we have $d_i \leq d_{L'} = \beta^{L'} \cdot (\alpha\beta) = O(L^4)$. The last equality follows from equation 57. Hence, from Corollary 5.3, we infer that $\deg(v, E_i) \leq d_i = O(L^4)$ for all levels $i < L'$ as well. Since $Y = \bigcup_{i=0}^{L'-1} E_i$, summing over all the levels $i \in [0,L]$, we infer that $\deg(v, X \cup Y) = O(L^5)$. □

Accordingly, we devote the rest of this section towards proving the second part of Theorem 5.8. But, before proceeding any further, we need to introduce the following notations.

- Let $V_X = \{v \in V : \ell(v) \geq L'\}$ denote the set of nodes lying at levels L' or above in the (α,β)-partition. Similarly, we let $V_Y = \{v \in V : \ell(v) < L'\}$ denote the set of nodes lying at levels $(L'-1)$ or below. Clearly, the node-set V is partitioned by these two subsets $V_X \subseteq V$ and $V_Y = V \setminus V_X$. Furthermore, note that the level of any edge (u,v) in the (α,β)-partition is defined as the maximum level among its endpoints. Hence, any edge $e \in Y = \bigcup_{i=0}^{L'-1} E_i$ has both of its endpoints in V_Y. On the other hand, any edge $e \in E_i$ with $i \geq L'$ has at least one of its endpoints in V_X. As a corollary, every edge $e \in X$ has at least one endpoint in V_X. We will use these observations throughout the rest of this section.

We will now try to explain why we consider these node-sets V_X and V_Y. For this we need to look back at the reason behind constructing the sets $\{X_i\}$ in the first place. Consider any node $v \in V_X$. Such a node belongs to a level $\ell(v) \geq L'$ in the (α,β)-partition. In an ideal scenario, for each $i \geq L'$, we want the degree of v to drop by a factor of (L^4/d_i) as we switch from the edge-set E_i to the edge-set $X_i \subseteq E_i$ (see the condition (5) in

Definition 5.6). If we succeed in this goal, then we can increase the weight of every edge in X_i by the inverse factor (d_i/L^4). As a consequence, the total weight received by the node v from the level-i edges will remain the same as before. To continue with the discussion, recall that every edge $e \in E_i$ had a weight $w(e) = 1/d_i$ in the (α,β)-partition. Let $w'(e) = w(e) \cdot (d_i/L^4) = 1/L^4$ denote the scaled up weight of the surviving edges $e \in X_i \subseteq E_i$. Thus, we expect to see that $W(v,E_i) = W'(v,X_i)$ at every level $i \in [L',L]$. Since $v \in V_X$, every edge incident upon v belongs to some E_i with $i \geq L'$. Accordingly, summing over all the levels $i \in [L',L]$, we expect to get the following guarantee.

$$W(v,E) = \sum_{i=L'}^{L} W(v,E_i) = \sum_{i=L'}^{L} W'(v,X_i) = W'(v,X) \geq (\alpha\beta)^{-1}.$$

The last inequality is supposed to hold due to Invariant 5.2 (recall that $\ell(v) \geq L'$). Thus, in an ideal scenario, if we assign a weight $w'(e) = L^{-4}$ to every edge $e \in X$, then we should get a fractional matching where every node in V_X gets a weight very close to one. In other words, such a matching will be "nearly maximal" in the graph induced by the edges $e \in E$ with $\ell(e) \geq L'$ (this holds since every such edge will have at least one endpoint in V_X). Thus, there exists a near-maximal fractional matching w' in the subgraph (V,X) that is also near-maximal in $(V, \bigcup_{i \geq L'} E_i)$. Since maximal matchings are 2-approximations to maximum matchings, intuitively, as we switch from the edge-set $\bigcup_{i \geq L'} E_i$ to the edge-set X, we loose roughly a factor of 2 in the approximation ratio (albeit in the fractional world). All the remaining edges $\bigcup_{i<L'} E_i$ go to the set Y, and we ought to loose nothing there in terms of the approximation guarantee. Thus, intuitively, there should be a matching defined on the edge-set $X \cup Y$ that is a good approximation to the maximum matching in the original graph $G = (V,E)$.

- It is now high time to emphasize that none of the above observations are exactly true to begin with. For example, as the Definition 5.6 states, there will be spurious nodes whose degrees might not drop by much as we go from from the edge-set E_i to the edge-set X_i, $i \in [L',L]$. There might be tiny nodes whose degrees might drop by more than we bargained for during the same transition. And even the degrees of the big, non-spurious nodes might not drop exactly by the factor (L^4/d_i). Nevertheless, we will try to carry out the plan outlined above, hoping to avoid all the roadblocks that we might come across while writing the formal proof.

Along these lines, we now formally define the weight function $w' : X \to [0,1]$ in equation 73.

$$w'(u,v) = h_0 \cdot (\lambda_{d_i} L^4)^{-1} \text{ for every edge } (u,v) \in X_i, i \in [L',L]. \tag{73}$$

Note that λ_{d_i} lies between $1/2$ and 1, and h_0 is very close to one (see equations 59, 69). Thus, although w' might assign different weights to different edges, none of these weights are too far away from $1/L^4$ (this is consistent with our preliminary discussion above). Not surprisingly, the next lemma shows that under w', every node $v \in V_X$ gets a weight that is lower bounded by a value very close to one. The proof of Lemma 6.2 appears in Section 6.2.

Lemma 6.2. *We have $W'(v,X) \geq h_1$ for all nodes $v \in V_X$. In other words, for every node v at level $\ell(v) \in [L',L]$ in the (α,β)-partition, the weight received by v under w' is very close to one.*

Unfortunately, however, the weights $\{w'(e)\}, e \in X$, do not give us a fractional matching. For example, consider any node $v \in V_X$ that is spurious at every level $i \geq L'$, i.e., $v \in S_i$ for all $i \in [L',L]$. At each level $i \geq L'$, the node v can have $\deg(v,X_i) = \Theta(L^4)$ (see the condition (6) in Definition 5.6), and its weight from this level $W'(v,X_i) = \deg(v,X_i) \cdot (h_0 \lambda_{d_i}^{-1}) \cdot L^{-4}$ can be as large as $\Theta(1)$. So the total weight it receives from all the levels can be as large as $(L - L') \cdot \Theta(1) = \Theta(L)$. The next lemma shows that this actually is the worst possible scenario. The proof of Lemma 6.3 appears in Section 6.3.

Lemma 6.3. *We have $W'(v,X) \leq 2L$ for every node $v \in V$.*

Since the spurious nodes are the ones that are preventing w' from being a fractional matching, we assign zero weight to any edge that is incident upon a spurious node, and leave the weights of the remaining edges as in w'. Accordingly, we define the weight function $w'' : X \to [0,1]$ in equation 74.

$$w''(u,v) = \begin{cases} w'(u,v) & \text{if } \{u,v\} \cap S = \emptyset, \text{ where } S = \bigcup_{i \geq L'} S_i; \\ 0 & \text{otherwise.} \end{cases} \tag{74}$$

We highlight an important fact here. Consider an edge $(u,v) \in X_i, i \geq L'$. It might very well be the case that one of its endpoints, say v, is non-spurious at level i but spurious at some other level $i' \in \{L', \ldots, L\} \setminus \{i\}$ (i.e., $v \notin S_i$ and $v \in S_{i'}$). Even under this situation, w'' assigns zero weight to the edge (u,v). Although this seems to be a stringent condition, it turns out not to affect our proof in any way.

As expected, the new weights $\{w''(e)\}, e \in X$, actually defines a fractional matching. This is shown in Lemma 6.4, and the proof of this lemma appears in Section 6.4.

Lemma 6.4. *We have $W''(v,X) \leq 1$ for all nodes $v \in V$. In other words, the weight function w'' defines a fractional matching in the subgraph induced by the edges in X.*

To take stock of the situation, we have constructed two weight functions $w' : X \to [0,1]$ and $w'' : X \to [0,1]$. The former nearly saturates all the nodes in V_X, in the sense that each such node receives a total weight that is at least η, for some η very close to one (see Lemma 6.2). But w' is not a fractional matching. The weight function w'', on the other hand, is a fractional matching (see Lemma 6.4) but might not saturate all the nodes in V_X. Our goal is to get the best of the both these worlds. As a first step towards this goal, in Lemma 6.5 we show that the nodes that are spurious at some level (a.k.a, the "troublemakers") are negligibly small in numbers compared to the size of the set V_X. The proof of Lemma 6.5 appears in Section 6.5.

Lemma 6.5. *The number of nodes that are spurious at some level is negligibly small compared to the number of nodes v at levels $\ell(v) \geq L'$ in the (α,β)-partition. Specifically, we have:*

$$|S| \leq (8\delta L^3/\varepsilon) \cdot |V_X|.$$

If we switch from the weight function w' to the weight function w'', the only difference we see is that the edges incident upon the nodes in S have been "turned off". Let these specific edges be called "transient". Under w', each node in S receives a total weight of at most $2L$ (see Lemma 6.3). Thus, we get an upper bound of $2L \cdot |S|$ on the total weight assigned by w' to all the transient edges. Since each edge is incident upon two nodes, as we switch from w' to w'' the total weight of the nodes in V_X drops by at most $4L \cdot |S|$. Consequently, by a simple counting argument, at most $(4L/\varepsilon) \cdot |S|$ many nodes in V_X can experience a larger than ε drop in their weights due to this transition. Since every node in V_X has weight at least h_1 under w' (see Lemma 6.2), we get the following lemma. The proof of Lemma 6.6 appears in Section 6.6.

Lemma 6.6. *Under w'', only a negligible fraction of the nodes in V_X do not receive large weights. Specifically, define the set of nodes $Q_X = \{v \in V_X : W''(v,X) < h_1 - \varepsilon\}$. Then we have:*

$$|Q_X| \leq (32\delta L^4/\varepsilon^2) \cdot |V_X|.$$

Corollary 6.7. *We have: $|V_X| \leq (1 - 32\delta L^4/\varepsilon^2)^{-1} \cdot |V_X \setminus Q_X|$.*

Proof. Follows from Lemma 6.6. □

Recall that every edge in $\bigcup_{i \geq L'} E_i$ has at least one endpoint in V_X. In lieu of our high level discussions at the start of this section, we see that w'' is a "nearly-maximal" fractional matching in $\bigcup_{i \geq L'} E_i$ in the following sense: It "nearly saturates" "almost all" the nodes in V_X (see Lemmas 6.4 and 6.6). Remaining faithful to our initial plan, in Lemma 6.8 we now show the existence of a large matching in $M^* \subseteq X \cup Y$. The construction of this matching M^* will crucially rely on the properties of the weight function w''. The proof of Lemma 6.8 appears in Section 6.7.

Lemma 6.8. *Let $X^* = \{(u,v) \in X : w''(u,v) > 0\}$ be the set of edges that receive nonzero weights under w'', and let $E^* = X^* \cup Y$. Define $G^* = (V, E^*)$ to be the subgraph of $G = (V,E)$ induced by the edges in E^*. Then for every matching $M \subseteq E$ in G, there is a matching $M^* \subseteq E^*$ in G^* such that $|M| \leq (2/h_2) \cdot (1+\varepsilon) \cdot |M^*|$.*

Theorem 5.8 now follows from Lemmas 6.1 and 6.8.

6.1 Some basic notations

Here, we quickly recall some basic notations introduced earlier. We also introduce a few notations. All these notations will be used throughout the rest of this section.

- Define $B = \bigcup_{i=L'}^{L} B_i$ to be the set of nodes that are big at some level $i \geq L'$.
- Define $S = \bigcup_{i=L'}^{L} S_i$ to be the set of nodes that are spurious at some level $i \geq L'$.
- Define $T = \bigcup_{i=L'}^{L} T_i$ to be the set of nodes that are tiny at some level $i \geq L'$.
- Given any node $v \in V$, let $\mathscr{L}_v(B)$ be the levels in $[L', L]$ where it is big. Thus, a level $i \in [L', L]$ belongs to the set $\mathscr{L}_v(B)$ iff $v \in B_i$.
- Given any node $v \in V$, let $\mathscr{L}_v(T)$ be the levels in $[L', L]$ where it is tiny. Thus, a level $i \in [L', L]$ belongs to the set $\mathscr{L}_v(T)$ iff $v \in T_i$.
- Given any node $v \in V$, let $\mathscr{L}_v(S)$ be the levels in $[L', L]$ where it is spurious. Thus, a level $i \in [L', L]$ belongs to the set $\mathscr{L}_v(S)$ iff $v \in S_i$.
- Since a node is either big or tiny (but not both) at each level, it follows that for every node $v \in V$ the set $\{L', \ldots, L\}$ is partitioned by the three subsets $\mathscr{L}_v(S)$, $\mathscr{L}_v(B) \setminus \mathscr{L}_v(S)$ and $\mathscr{L}_v(T) \setminus \mathscr{L}_v(S)$.
- Recall the notations V_X and V_Y introduced right after the proof of Lemma 6.1.

6.2 Proof of Lemma 6.2

- Throughout this proof, we will use the notations defined in Section 6.1.

Throughout this section, fix any node $v \in V_X$. Thus, the node v is at level $\ell(v) \geq L'$ in the (α, β)-partition.

Claim 6.9. *Under w, the node v gets negligible weight from all the levels $i \in [L', L]$ where it is tiny. Specifically, we have:*

$$\sum_{i \in \mathscr{L}_v(T)} W(v, X_i) \leq 3\varepsilon.$$

Proof. Consider any level $i \in \mathscr{L}_v(T)$ where the node v is tiny. By the condition (2) in Definition 5.6, we have $\deg(v, E_i) < 3\varepsilon d_i / L^2$. Since each edge in E_i received a weight $1/d_i$ under w, we get:

$$W(v, E_i) = \deg(v, E_i) \cdot (1/d_i) \leq 3\varepsilon / L^2 \tag{75}$$

Since there are at most L levels in the range $[L', L]$, summing equation 75 over all $i \in \mathscr{L}_v(T)$, we get:

$$\sum_{i \in \mathscr{L}_v(T)} W(v, E_i) \leq |\mathscr{L}_v(T)| \cdot (3\varepsilon / L^2) \leq 3\varepsilon / L \leq 3\varepsilon.$$

The last inequality holds since $L \geq 1$ (see equation 65). □

Corollary 6.10. *Under w, the node v receives close to one weight from the levels $i \in [L', L]$ where it is big. Specifically, we have:*

$$\sum_{i \in \mathscr{L}_v(B)} W(v, E_i) \geq (\alpha\beta)^{-1} - 3\varepsilon.$$

Proof. Since $v \in V_X$, the node v belongs to a level $\ell(v) \geq L'$ in the (α, β)-partition. Hence, every edge $(u,v) \in E$ incident upon v has level $\ell(u,v) = \max(\ell(u), \ell(v)) \geq L'$. Thus, from Invariant 5.2 we get:

$$\sum_{i \in L'}^{L} W(v, E_i) = W(v, E) \geq (\alpha\beta)^{-1} \tag{76}$$

At any level in the (α,β)-partition the node v is either big or tiny, but it cannot be both at the same time. In other words, the set of levels $\{L',\ldots,L\}$ is partitioned into two subsets: $\mathscr{L}_v(B)$ and $\mathscr{L}_v(T)$. Hence, the corollary follows from equation 76 and Claim 6.9. □

Claim 6.11. *Consider the weight received by the node v under w' from the levels in $[L',L]$ where it is big. This weight is nearly equal to the weight it receives under w from the same levels. Specifically, we have:*

$$\sum_{i\in\mathscr{L}_v(B)} W'(v,X_i) \geq (h_0 e^{-\gamma})\cdot \sum_{i\in\mathscr{L}_v(B)} W(v,E_i).$$

Proof. Consider any level $i\in\mathscr{L}_v(B)$ where the node v is big. By the condition (5) of Definition 5.6, the degree of v drops roughly by a multiplicative factor of $(\lambda_{d_i}L^4/d_i)$ as we go from E_i to X_i. Thus, we have:

$$\deg(v,X_i) \geq e^{-\gamma}\cdot(\lambda_{d_i}L^4/d_i)\cdot\deg(v,E_i) \tag{77}$$

Each edge in X_i receives exactly $h_0\cdot(\lambda_{d_i}L^4)^{-1}$ weight under w'. Hence, equation 77 implies that:

$$W'(v,X_i) = h_0\cdot(\lambda_{d_i}L^4)^{-1}\cdot\deg(v,X_i) \geq (h_0e^{-\gamma})\cdot(1/d_i)\cdot\deg(v,E_i) \tag{78}$$

Since each edge in E_i receives $1/d_i$ weight under w, equation 78 implies that:

$$W'(v,X_i) \geq (h_0e^{-\gamma})\cdot W(v,E_i) \tag{79}$$

The claim follows if we sum equation 79 over all levels $i\in\mathscr{L}_v(B)$. □

Under w', the total weight received by the node v is at least the weight it receives from the levels $i\in[L',L]$ where it is big. Thus, from Corollary 6.10 and Claim 6.11, we infer that:

$$W'(v,X) \geq \sum_{i\in\mathscr{L}_v(B)} W'(v,X_i) \geq (h_0e^{-\gamma})\cdot((\alpha\beta)^{-1}-3\varepsilon) = h_1.$$

The last equality follows from equation 71. This concludes the proof of the lemma.

6.3 Proof of Lemma 6.3

Fix any node $v\in V$. We will first bound the weight received by the node v under w' from any given level $i\in[L',L]$. Towards this end, note that by the condition (6) in Definition 5.6, we have $\deg(v,X_i)\leq \lambda_{d_i}\cdot L^4+2\leq 2\lambda_{d_i}L^4$. The last inequality holds since $1/2\leq\lambda_{d_i}\leq 1$ (see equation 59) and $L^4\geq 4$ (see equation 65). Since every edge in X_i receives a weight $h_0\cdot(\lambda_{d_i}L^4)^{-1}$ under w', equation 70 gives us:

$$W'(v,X_i) = \deg(v,X_i)\cdot h_0\cdot(\lambda_{d_i}L^4)^{-1} \leq 2h_0 \leq 2$$

Summing the above inequality over all levels $i\in[L',L]$, we get:

$$W'(v,X) = \sum_{i=L'}^{L} W'(v,X_i) \leq 2(L-L'+1) \leq 2L.$$

This concludes the proof of the lemma.

6.4 Proof of Lemma 6.4

- Throughout this proof, we will use the notations defined in Section 6.1.

Throughout this section, we fix any node $v \in V$. Recall that the set of levels $\{L', \ldots, L\}$ is partitioned into three subsets: $\mathscr{L}_v(S)$, $\mathscr{L}_v(T) \setminus \mathscr{L}_v(S)$ and $\mathscr{L}_v(B) \setminus \mathscr{L}_v(S)$. Thus, we have:

$$W''(v,X) = \sum_{i \in \mathscr{L}_v(S)} W''(v,X_i) + \sum_{i \in \mathscr{L}_v(T) \setminus \mathscr{L}_v(S)} W''(v,X_i) + \sum_{i \in \mathscr{L}_v(B) \setminus \mathscr{L}_v(S)} W''(v,X_i) \tag{80}$$

We separately bound the weights received by the node v under w'' from these three different types of levels. The lemma follows by adding up the bounds from Claims 6.12, 6.13 and 6.14.

Claim 6.12. *The node v gets zero weight under w'' from all the levels $i \in [L', L]$ where it is spurious. Specifically, we have $\sum_{i \in \mathscr{L}_v(S)} W''(v,E_i) = 0$.*

Proof. If the node v is spurious at some level i, then each of its incident edges in X_i gets zero weight under w'' (see equation 74). The claim follows. □

Claim 6.13. *The node v gets negligible weight under w'' from all the levels $i \in [L', L]$ where it is tiny but non-spurious. Specifically, we have:*

$$\sum_{i \in \mathscr{L}_v(T) \setminus \mathscr{L}_v(S)} W''(v,X_i) \leq (1+4\varepsilon)^{-1} \cdot (4\varepsilon).$$

Proof. Consider any level $i \in \mathscr{L}_v(T) \setminus \mathscr{L}_v(S)$ where the node v is tiny but non-spurious. Note that $\lambda_{d_i} \geq 1/2$ (see equation 59). Hence, Definition 5.6 and equation 63 imply that:

$$\deg(v,X_i) \leq 3\varepsilon\lambda_{d_i}L^2 + 2 \leq 4\varepsilon\lambda_{d_i}L^2 \tag{81}$$

Each edge in X_i receives at most $h_0 \cdot (\lambda_{d_i}L^4)^{-1}$ weight under w''. Thus, equations 81 and 70 imply that:

$$W''(v,X_i) \leq h_0 \cdot (\lambda_{d_i}L^4)^{-1} \cdot \deg(v,X_i) \leq 4\varepsilon h_0 L^{-2} \leq (1+4\varepsilon)^{-1} \cdot (4\varepsilon/L^2) \tag{82}$$

Since there are at most L levels in the range $[L', L]$, summing equation 82 over all $i \in \mathscr{L}_v(T) \setminus \mathscr{L}_v(S)$ gives:

$$\sum_{i \in \mathscr{L}_v(T) \setminus \mathscr{L}_v(S)} W''(v,X_i) \leq (1+4\varepsilon)^{-1} \cdot (4\varepsilon/L) \tag{83}$$

This claim follows from equation 83 and the fact that $L \geq 1$ (see equation 65). □

Claim 6.14. *The node v receives at most $(1+4\varepsilon)^{-1}$ weight under w'' from all the levels $i \in [L', L]$ where it is big and non-spurious. Specifically, we have:*

$$\sum_{i \in \mathscr{L}_v(B) \setminus \mathscr{L}_v(S)} W''(v,X_i) \leq (1+4\varepsilon)^{-1}.$$

Proof. Consider any level $i \in \mathscr{L}_v(B) \setminus \mathscr{L}_v(S)$ where the node v is big but non-spurious. By the condition (5) of Definition 5.6, the degree of v drops roughly by a multiplicative factor of $(\lambda_{d_i}L^4/d_i)$ as we go from E_i to X_i. Specifically, we have:

$$\deg(v,X_i) \leq e^{\gamma} \cdot (\lambda_{d_i}L^4/d_i) \cdot \deg(v,E_i) \tag{84}$$

Each edge in X_i receives at most $h_0 \cdot (\lambda_{d_i}L^4)^{-1}$ weight under w''. Hence, equations 69, 84 imply that:

$$W''(v,X_i) \leq h_0 \cdot (\lambda_{d_i}L^4)^{-1} \cdot \deg(v,X_i) \leq (1+4\varepsilon)^{-1} \cdot (1/d_i) \cdot \deg(v,E_i) \tag{85}$$

Since each edge in E_i receives $1/d_i$ weight under w, equation 85 implies that:

$$W''(v,X_i) \leq (1+4\varepsilon)^{-1} \cdot W(v,E_i) \tag{86}$$

The claim follows if we sum equation 86 over all levels $i \in \mathscr{L}_v(B) \setminus \mathscr{L}_v(S)$, and recall that the sum $\sum_{i \mathscr{L}_v(B) \setminus \mathscr{L}_v(S)} W(v,E_i)$ itself is at most one (for w is a fractional matching in G). □

The lemma follows from equation 80 and Claims 6.12, 6.13, 6.14.

6.5 Proof of Lemma 6.5

- Throughout this proof, we will use the notations defined in Section 6.1.

We first show that the number of nodes that are spurious at some level is negligibly small compared to the number of nodes that are big at some level. Towards this end, recall the condition (4) in Definition 5.6. For each level $i \in [L', L]$, this implies that $|S_i| \leq 4\delta \cdot |B_i| \leq 4\delta \cdot |B|$. The last inequality holds since $B_i \subseteq B$. Hence, summing over all the levels $i \in [L', L]$, we get:

$$|S| = \left| \bigcup_{i=L'}^{L} S_i \right| \leq \sum_{i=L'}^{L} |S_i| \leq (L - L' + 1) \cdot 4\delta \cdot |B| \leq (4\delta L) \cdot |B| \tag{87}$$

It remains to upper bound the size of the set B in terms of the size of the set V_X. Towards this end, we first show that every node in B receives sufficiently large weight under w. Specifically, fix any node $v \in B$. By definition, we have $v \in B_i$ at some level $i \in [L', L]$. Hence, the condition (1) in Definition 5.6 implies that $\deg(v, E_i) > \varepsilon d_i / L^2$. Since each edge in E_i receives a weight $1/d_i$ under w, we get: $W(v, E_i) = (1/d_i) \cdot \deg(v, E_i) > \varepsilon / L^2$. To summarize, we have the following guarantee.

$$\sum_{i=L'}^{L} W(v, E_i) \geq \varepsilon / L^2 \quad \text{for all nodes } v \in B. \tag{88}$$

Recall that the level of an edge (u, v) in the (α, β)-partition is given by $\ell(u, v) = \max(\ell(u), \ell(v))$. Further, the set E_i is precisely the set of those edges $e \in E$ with $\ell(e) = i$. Hence, each edge $(u, v) \in \bigcup_{i=L'}^{L} E_i$ has at least one endpoint in $V_X = \{x \in V : \ell(v) \in [L', L]\}$. Thus, a simple counting argument gives us:

$$\sum_{v \in B} \sum_{i=L'}^{L} W(v, E_i) \leq 2 \cdot \sum_{v \in V_X} \sum_{i=L'}^{L} W(v, E_i) \tag{89}$$

The above inequality holds for the following reason. Consider any edge $(u, v) \in \bigcup_{i=L'}^{L} E_i$. Either both the endpoints u, v belong to V_X, or exactly one of the endpoints u, v belong to V_X. In the former case, the edge (u, v) contributes at most $2 \cdot w(u, v)$ to the left hand side, and exactly $4 \cdot w(u, v)$ to the right hand side. In the latter case, the edge (u, v) contributes at most $2 \cdot w(u, v)$ to the left hand side, and exactly $2 \cdot w(u, v)$ to the right hand side. Thus, the contribution of every relevant edge to the left hand side is upper bounded by its contribution to the right hand side.

Next, recall that w defines a fractional matching in G, and so we have $W(v, E) \leq 1$ for every node $v \in V$. Accordingly, from equations 88 and 89, we infer that:

$$|B| \cdot (\varepsilon / L^2) \leq \sum_{v \in B} \sum_{i=L'}^{L} W(v, E_i) \leq 2 \cdot \sum_{v \in V_X} \sum_{i=L'}^{L} W(v, E_i) \leq 2 \cdot |V_X|.$$

Rearranging the terms of the above inequality, we get our desired upper bound on $|B|$.

$$|B| \leq (2L^2 / \varepsilon) \cdot |V_X| \tag{90}$$

Finally, from equations 87, 90, we infer that $|S| \leq (8\delta L^3 / \varepsilon) \cdot |V_X|$. This concludes the proof of the lemma.

6.6 Proof of Lemma 6.6

- Throughout this proof, we will use the notations V_X and V_Y defined right after the proof of Lemma 6.1.

The weights w'' are constructed from w' by switching off the nodes in S and the edges incident upon them (see equations 73, 74). Since each node in S has at most $2L$ weight under w' (see Lemma 6.3), and since each edge is incident upon two nodes, the transition from w' to w'' decreases the sum of the node-weights in V_X by at most $4L \cdot |S|$. This insight is formally stated in the claim below.

Claim 6.15. *We have:*

$$\sum_{v \in V_X} W''(v,X) \geq \sum_{v \in V_X} W'(v,X) - 4L \cdot |S|.$$

Proof. To prove the claim, we first show that:

$$\sum_{v \in V_X} W''(v,X) + 2 \cdot \sum_{v \in S} W'(v,X) \geq \sum_{v \in V_X} W'(v,X) \tag{91}$$

Equation 91 follows from a simple counting argument. Consider any edge $(u,v) \in X$ that has at least one endpoint in V_X. These are the edges that contribute towards the right hand side of the above inequality. Now, there are two possible cases to consider.

- *Case 1.* At least one of the endpoints u, v belong to S. In this case, we have $w''(u,v) = 0$. However, due to the term $2 \cdot \sum_{v \in S} W'(v,S)$, the edge (u,v) contributes at least $2w'(u,v)$ towards the left hand side. And clearly, the edge (u,v) can contribute at most $2w'(u,v)$ towards the right hand side.
- *Case 2.* None of the endpoints u, v belong to S. In this case, we have $w'(u,v) = w''(u,v)$, and so the contribution of the edge (u,v) towards the left hand side is at least as much as its contribution towards the right hand side.

Next, we recall that $W'(v,X) \leq 2L$ for every node $v \in V$ (see Lemma 6.3). Thus, we have:

$$4L \cdot |S| \geq 2 \cdot \sum_{v \in S} W'(v,X) \tag{92}$$

From equations 91 and 92, we infer that:

$$\sum_{v \in V_X} W''(v,X) + 4L \cdot |S| \geq \sum_{v \in V_X} W'(v,X) \tag{93}$$

The claim follows from equation 93. □

As we make a transition from w' to w'', the total weight of the nodes in V_X drops by at most $4L \cdot |S|$ (see Claim 6.15). Hence, by a simple counting argument, due to this transition at most $(4L/\varepsilon) \cdot |S|$ nodes can experience their weights dropping by more than ε. Next, recall that under w', every node $v \in V_X$ has weight $W'(v,X) \geq h_1$ (see Lemma 6.2). Thus, every node in Q_X has experienced a drop of ε due to the transition from w' to w''. Accordingly, the size of the set Q_X cannot be larger than $(4L/\varepsilon) \cdot |S|$. Since $|S| \leq (8\delta L^3/\varepsilon) \cdot |V_X|$, we infer that $|Q_X| \leq (32\delta L^4/\varepsilon^2) \cdot |V_X|$. This concludes the proof of the lemma.

6.7 Proof of Lemma 6.8

- Throughout this proof, we will use the notations V_X and V_Y defined right after the proof of Lemma 6.1.

We begin by noting that under w'', the weights of the edges in X^* are very close to one another.

Observation 6.16. *Let $c = \lfloor (L^4/2) \rfloor$. Then we have $1/(8c) \leq w''(e) \leq 1/c$ for every edge $e \in X^*$.*

Proof. Consider any edge $(u,v) \in X^*$. This edge belongs to some level $i \in [L', L]$, and since $(u,v) \in X^*$, by definition the edge has nonzero weight under w''. Hence, equations 73 and 74 implies that $w''(u,v) = (h_0/\lambda_{d_i}) \cdot L^{-4}$. Since $1/2 \leq h_0, \lambda_{d_i} \leq 1$ (see equations 59, 70), we get: $1/(2L^4) \leq w''(u,v) \leq 2/L^4$. The observation now follows from the fact that $c = \lfloor (L^4/2) \rfloor \geq (L^4/4)$. See equation 54. □

As an important corollary, we get an upper bound on the maximum degree of a node in $G_{X^*} = (V, X^*)$.

Corollary 6.17. *We have $deg(v, X^*) \leq 8c$ for every node $v \in V$.*

Proof. The corollary holds since $1 \geq W''(v,X) = W''(v,X^*) \geq \deg(v,X^*) \cdot (1/c)$ for all nodes $v \in V$. The first inequality follows from Lemma 6.4. The last inequality follows from Observation 6.16. □

6.7.1 Outline of the proof

Before proceeding any further, we give a high level overview of our approach. Suppose that we make two simplifying assumptions (which will be relaxed in Section 6.7.2).

Assumption 1. *Each edge $e \in X^*$ receives exactly the same weight $1/c''$ under w'', for some integer c''. Thus, we have $w''(e) = 1/c''$ for all $e \in X^*$. By Observation 6.16, the weights $\{w''(e)\}, e \in X^*$, are already very close to one another. Here, we take this one step further by assuming that they are exactly equal. We also assume that these edge-weights are inverse of some integer value.*

Assumption 2. *There is no edge $(u,v) \in E$ in the (α,β)-partition at level $\ell(u,v) < L'$. In other words, we are assuming that the edge-set Y is empty.*

By Lemma 6.4, w'' defines a fractional matching in the graph $G = (V,E)$. We will now see that under the above two assumptions, we can say something more about w''. Towards this end, first note that as $Y = \emptyset$ (see Observation 2), every edge $(u,v) \in E$ has level $\ell(u,v) = \max(\ell(u),\ell(v)) \geq L'$ in the (α,β)-partition. Since V_X is precisely the set of nodes $v \in V$ with levels $\ell(v) \geq L'$, we infer the following fact.

Fact 1. *Every edge $(u,v) \in E$ has at least one endpoint in V_X.*

On the other hand, by Lemma 6.6, an overwhelming fraction of the nodes in V_X receive weights that are close to one under w''. This, along with Fact 1, implies that w'' is a "near-maximal" matching in G, in the sense that almost all the edges in G have at least one nearly tight endpoint under w''. Hence, the value of w'', given by $w''(E) = \sum_{e \in E} w''(e)$, is very close to being a 2-approximation to the size of the maximum cardinality matching in G. We will now show that there exists a matching $M^* \subseteq X^*$ whose size is very close to $w''(E)$. This will imply the desired guarantee we are trying to prove.

We now bound the value of the fractional matching w''. Since each edge in X^* receives $1/c''$ weight under w'' (see Assumption 1), and since every other edge gets zero weight, we have $w''(E) = |X^*|/c''$. This is summarized as a fact below.

Fact 2. *We have $w''(E) = |X^*|/c''$.*

It remains to show the existence of a large matching in G_{X^*} of size very close to $w''(E)$. Towards this end, we first note that c'' is an upper bound on the maximum degree of a node in $G_{X^*} = (V,X^*)$. This holds since $W''(v,X^*) = (1/c'') \cdot \deg(v,X^*) \leq 1$ for all $v \in V$ (see Lemma 6.4). Thus, by Vizing's theorem [17, 18], there is a proper edge-coloring of the edges in G_{X^*} using only $c''+1$ colors. Recall that a proper edge-coloring assigns one color to every edge in the graph, while ensuring that the edges incident upon the same node get different colors. We take any $c''+1$ edge-coloring in G_{X^*} as per Vizing's theorem. By a simple counting argument, there exists a color that is assigned to at least $|X^*|/(c''+1)$ edges in X^*. The edges that receive this color constitute a matching in G_{X^*} (for they cannot share a common endpoint). In other words, there exists a matching $M^* \subseteq X^*$ of size $|X^*|/(c''+1)$. From Fact 2, we infer that $|M^*| \geq w''(E) \cdot (c''/(c''+1))$. Since $c'' = \Theta(\text{poly } L) = \Theta(\text{poly} \log n)$, the size of the matching M^* is indeed very close to the value $w''(E)$. This gives us the desired bound.

6.7.2 The complete proof

Unfortunately, while giving a complete proof, we cannot rely on the simplifying assumptions from Section 6.7.1. Since Assumption 1 no longer holds, a natural course of action is to discretize the edge-weights under w'' into constantly many buckets, without loosing too much of the value of the fractional matching defined by w''. This will at least ensure that the number of different weights is some constant, and we hope to extend our proof technique from Section 6.7.1 to this situation.

We pick some sufficiently large constant integer K, and round down the weights under w'' to the nearest multiples of $1/(8Kc)$. Let w^* be these rounded weights. Specifically, for every edge $e \in X^*$, we have:

$$w^*(e) \quad = \quad \frac{\tau^*(e)}{8Kc}, \text{ where } \tau^*(e) \text{ is a positive integer such that } \frac{\tau^*(e)}{8Kc} \leq w''(e) < \frac{\tau^*(e)+1}{8Kc}. \tag{94}$$

As usual, given any subset of edges $E' \subseteq E$ and node $v \in V$, we define $W^*(v,E')$ to be the total weight received by v from its incident edges in E', and $w^*(E')$ to be the total weight of all the edges in E'. We now list down some basic properties of the weights w^*, comparing them against the weights w''.

Observation 6.18. *For all $e \in X^*$, we have $(8c)^{-1} \leq w^*(e) \leq w''(e) \leq (c)^{-1}$ and $K \leq \tau^*(e) \leq 8K$.*

Proof. Follows from Observation 6.16 and equation 94. □

Observation 6.19. *For every node $v \in V$, we have $W''(v,X^*) - 1/K \leq W^*(v,X^*) \leq W''(v,X^*)$.*

Proof. Fix any node $v \in V$. It has at most $8c$ edges incident upon it in $G_{X^*} = (V,X^*)$ (see Corollary 6.17). As we make a transition from w'' to w^*, each of these edges incurs a weight-loss of at most $1/(8Kc)$ (see equation 94). Hence, the total loss in the weight received by the node v from its incident edges is at most $(8c) \cdot 1/(8Kc) = 1/K$. Accordingly, we have $W^*(v,X^*) \geq W''(v,X^*) - 1/K$. □

Roadmap. The rest of the section is organized as follows.

1. Recall that $M \subseteq E$ is a matching in the graph $G = (V,E)$ we want to compete against. We first construct two multi-graphs $\mathscr{G}_{X^*} = (V, \mathscr{E}_{X^*})$ and $\mathscr{G}_{M \cap Y} = (V, \mathscr{E}_{M \cap Y})$. The multi-edges in $\mathscr{E}_{X^*}$ are constructed from the edge-set X^*. Similarly, as the notation suggests, the multi-edges in $\mathscr{E}_{M \cap Y}$ are constructed from the edge-set $M \cap Y$ (which consists of the matched edges in M with both endpoints at a level less than L' in the (α,β)-partition). We also define $\mathscr{G} = (V, \mathscr{E})$ to be the union of the two multi-graphs $\mathscr{G}_{X^*}$ and $\mathscr{G}_{M \cap Y}$, so that $\mathscr{E} = \mathscr{E}_{X^*} \cup \mathscr{E}_{M \cap Y}$.
2. Next, we construct a fractional matching $w_{\mathscr{E}} : \mathscr{E} \to [0,1]$ in the multi-graph $\mathscr{G}$. This fractional matching is "uniform", in the sense that it assigns exactly the same weight to every multi-edge. This is a significant property, since the fractional matching we had to deal with in the simplified proof of Section 6.7.1 was also uniform, and this fact was used while comparing the value of the fractional matching against the size of the integral matching constructed out of Vizing's theorem [17, 18]. As in Section 6.7.1, we will show that the value of the fractional matching $w_{\mathscr{E}}(\mathscr{E}) = \sum_{e \in \mathscr{E}} w_{\mathscr{E}}(e)$ is very close to being a 2-approximation to the size of $M \subseteq E$ (the matching in G we are competing against).
3. Finally, we construct a matching $M^* \subseteq X^* \cup Y$ whose size is very close to $w_{\mathscr{E}}(\mathscr{E})$. To construct this matching M^*, we use a generalized version of Vizing's theorem for multi-graphs.
4. Steps 2 and 3 imply that the size of the matching M^* is very nearly within a factor of 2 of $|M|$. This concludes the proof of Lemma 6.8.

Step I: Constructing the multi-graphs $\mathscr{G}_{X^*} = (V, \mathscr{E}_{X^*})$, $\mathscr{G}_{M \cap Y} = (V, \mathscr{E}_{M \cap Y})$ and $\mathscr{G} = (V, \mathscr{E})$.

We create $\tau^*(e)$ copies of each edge $e \in X^*$, and let $\mathscr{E}_{X^*}$ denote the collection of these multi-edges. Next, we create $\max(0, 8Kc - \deg(u, \mathscr{E}_{X^*}) - \deg(v, \mathscr{E}_{X^*}))$ copies of every edge $(u,v) \in M \cap Y$, and let $\mathscr{E}_{M \cap Y}$ denote the collection of these multi-edges. We also define $\mathscr{E} = \mathscr{E}_{X^*} \cup \mathscr{E}_{M \cap Y}$. We will be interested in the multi graphs $\mathscr{G}_{X^*} = (V, \mathscr{E}_{X^*})$, $\mathscr{G}_{M \cap Y} = (V, \mathscr{E}_{M \cap Y})$ and $\mathscr{G} = (V, \mathscr{E})$. We now state three simple observations that will be useful later on.

Observation 6.20. *In the multi-graph $\mathscr{G}$, the degree of every node $v \in V$ is at most $8Kc$.*

Proof. Throughout the proof, fix any node $v \in V$. We first bound the degree of this node among the multi-edges in $\mathscr{E}_{X^*}$. Accordingly, consider an edge $(u,v) \in X^*$ that contributes to $\deg(v, \mathscr{E}_{X^*})$. Under w^*, this edge has weight $w^*(u,v) = \tau^*(u,v)/(8Kc)$. While constructing the multi-graph $\mathscr{G}$, we simply created $\tau^*(u,v)$ copies of this edge. Thus, summing over all such edges $(u,v) \in X^*$, we get:

$$\begin{aligned} \deg(v, \mathscr{E}_{X^*}) &= \sum_{(u,v)\in X^*} \tau^*(u,v) \\ &= \sum_{(u,v)\in X^*} w^*(u,v) \cdot (8Kc) \\ &= W^*(v, X^*) \cdot (8Kc) \\ &\leq 8Kc \end{aligned} \tag{95}$$

The last inequality holds since $X^* \subseteq X$ is the set of edges in X that receive nonzero weights under w'', and hence, we have $W^*(v,X^*) \leq W''(v,X^*) = W''(v,X) \leq 1$ (see Observation 6.19 and Lemma 6.4).

Now, we have two cases to consider.

Case 1. $v \in V_X$. In this case, all the multi-edges in $\mathscr{G}$ that are incident upon it originate from the edge-set X^*. Thus, we have $\deg(v, \mathscr{E}) = \deg(v, \mathscr{E}_{X^*})$. The observation now follows from equation 95.

Case 2. $v \in V_Y$. In this case, note that the node v can be incident upon at most one edge in $M \cap Y$ (for M is a matching). Thus, by our construction of the multigraph, at most $(8Kc - \deg(v, \mathscr{E}_{X^*}))$ multi-edges incident upon v are included in $\mathscr{E}_{M\cap Y}$. We therefore conclude that:

$$\deg(v, \mathscr{E}) = \deg(v, \mathscr{E}_{X^*}) + \deg(v, \mathscr{E}_{M\cap Y}) \leq \deg(v, \mathscr{E}_{X^*}) + 8Kc - \deg(v, \mathscr{E}_{X^*}) = 8Kc.$$

□

Observation 6.21. *In the multi-graph $\mathscr{G}_{X^*}$, there are at most $8K$ multi-edges joining any two given nodes.*

Proof. Consider any two nodes $u, v \in V$. If $(u,v) \notin X^*$, then by our construction, there cannot be any multi-edge between u and v in $\mathscr{G}_{X^*}$. Hence, we assume that $(u,v) \in X^*$. Recall that $w^*(u,v) = \tau^*(u,v)/(8Kc)$ and that $\tau^*(u,v)$ copies of the edge (u,v) are added to the multi-graph $\mathscr{G}_{X^*}$. Thus, it remains to show that $\tau^*(u,v) \leq 8K$, which clearly holds since $w^*(u,v) \leq 1/c$ (see Observation 6.18). □

Observation 6.22. *In multi-graph $\mathscr{G}$, we have $deg(u, \mathscr{E}) + deg(v, \mathscr{E}) \geq 8Kc$ for each edge $(u,v) \in M \cap Y$.*

Proof. If $\deg(u, \mathscr{E}_{X^*}) + \deg(v, \mathscr{E}_{X^*}) \geq 8Kc$, then there is nothing more to prove. So let $\deg(u, \mathscr{E}_{X^*}) + \deg(v, \mathscr{E}_{X^*}) = 8Kc - \mu$ for some integer $\mu \geq 1$. In this case, by our construction, μ copies of the edge (u,v) get added to $\mathscr{E}_{M\cap Y}$, and hence to $\mathscr{E} = \mathscr{E}_{X^*} \cup \mathscr{E}_{M\cap Y}$. Thus, we again get: $\deg(u, \mathscr{E}) + \deg(v, \mathscr{E}) \geq 8Kc$. □

Step II: Constructing the fractional matching $w_{\mathscr{E}} : \mathscr{E} \to [0,1]$ in $\mathscr{G}$.

We assign a weight $w_{\mathscr{E}}(e) = (8Kc)^{-1}$ to every multi-edge $e \in \mathscr{E}$. Since every node in $\mathscr{G}$ has degree at most $8Kc$, we infer that:

$$W_{\mathscr{E}}(v, \mathscr{E}) = \deg(v, \mathscr{E}) \cdot 1/(8Kc) \leq 1.$$

This shows that the weights $\{w_{\mathscr{E}}(e)\}, e \in \mathscr{E}$, constitute a fractional matching in the multi-graph $\mathscr{G}$. We now state an easy bound on the value of this fractional matching.

Observation 6.23. *We have $\sum_{e\in\mathscr{E}} w_{\mathscr{E}}(e) = |\mathscr{E}|/(8Kc)$.*

As an aside, we intimate the reader that in Step III we will construct a matching $M^* \subseteq X^* \cup Y$ of size at least $|\mathscr{E}|/(8K(c+1))$. Thus, Observation 6.23 will imply that the size of M^* is a $(1+1/c)$-approximation to the value of $w_{\mathscr{E}}$. At the present moment, however, our goal is to compare the value of $w_{\mathscr{E}}$ against the size of the matching M. We now make two simple observations that will be very helpful in this regard.

Observation 6.24. *For every edge $(u,v) \in M \cap Y$, we have $W_{\mathscr{E}}(u) + W_{\mathscr{E}}(v) \geq 1$.*

Proof. Fix any edge $(u,v) \in M \cap Y$. Since $(u,v) \in Y$, both u and v are at levels less than L' in the (α,β)-partition. This means that $u,v \notin V_X$. So there can be no multi-edge in $\mathscr{E}_{X^*}$ between u and v. We consider two possible cases.

- *Case 1.* $\deg(u,\mathscr{E}_{X^*}) + \deg(v,\mathscr{E}_{X^*}) \geq 8Kc$. Note that every multi-edge in $\mathscr{E}_{X^*}$ gets a weight $1/(8Kc)$ under $w_{\mathscr{E}}$. Since no such multi-edge joins u and v, we have:

$$\begin{aligned} W^*(u,\mathscr{E}) + W^*(v,\mathscr{E}) &\geq W^*(u,\mathscr{E}_{X^*}) + W^*(v,\mathscr{E}_{X^*}) \\ &= (\deg(u,\mathscr{E}_{X^*}) + \deg(v,\mathscr{E}_{X^*})) \cdot (8Kc)^{-1} \\ &\geq 1 \end{aligned}$$

- *Case 2.* $\deg(u,\mathscr{E}_{X^*}) + \deg(v,\mathscr{E}_{X^*}) = 8Kc - \mu$, where $\mu > 0$ (say). In this case, we also make μ copies of the edge (u,v) and include them in $\mathscr{E}_{M\cap Y}$. Note that every multi-edge in $\mathscr{E} = \mathscr{E}_{X^*} \cup \mathscr{E}_{M\cap Y}$ gets a weight $1/(8Kc)$ under $w_{\mathscr{E}}$. Since there is no multi-edge between u and v in $\mathscr{E}_{X^*}$, we get:

$$\begin{aligned} W^*(u,\mathscr{E}) + W^*(v,\mathscr{E}) &\geq W^*(u,\mathscr{E}_{X^*}) + W^*(v,\mathscr{E}_{X^*}) + W^*(u,\mathscr{E}_{M\cap Y}) \\ &= (\deg(u,\mathscr{E}_{X^*}) + \deg(v,\mathscr{E}_{X^*}) + \mu) \cdot (8Kc)^{-1} \\ &\geq 1 \end{aligned}$$

□

Observation 6.25. *We have $W_{\mathscr{E}}(v) \geq h_1 - \varepsilon - (1/K)$ for every node $v \in V_X \setminus Q_X$ (see Lemma 6.6).*

Proof. Consider any node $v \in V_X \setminus Q_X$. Consider any edge $(u,v) \in X^*$ incident upon v. This edge has weight $w^*(u,v) = \tau^*(u,v)/(8Kc)$ under w^* (see equation 94). Under $w_{\mathscr{E}}$, on the other hand, we create $\tau^*(u,v)$ copies of this edge and assign a weight $1/(8Kc)$ to each of these copies. The total weight assigned to all these copies, therefore, remains exactly equal to $\tau^*(u,v)/(8Kc)$. Next, note that since $v \in V_X$, there cannot be any edge in Y that is incident upon v. Thus, we have:

$$W_{\mathscr{E}}(v,\mathscr{E}) = W_{\mathscr{E}}(v,\mathscr{E}_{X^*}) = W^*(v,X^*) \geq W''(v,X^*) - (1/K) \geq h_1 - \varepsilon - (1/K).$$

The second-last inequality follows from Observation 6.19. The last inequality follows from Lemma 6.6. □

We will now bound the sum of the node-weights under $w_{\mathscr{E}}$ by the size of the matching M.

Claim 6.26. *We have $|M| \leq (1/h_2) \cdot \sum_{v\in V} W_{\mathscr{E}}(v,\mathscr{E})$.*

Proof. Every edge $(u,v) \in M \setminus Y$ has at least one endpoint at level L' or higher in the (α,β)-partition. In other words, every edge $(u,v) \in M \setminus Y$ has at least one endpoint in V_X. Since each node in V_X can get matched by at most one edge in M, we have:

$$|M \setminus Y| \leq |V_X| \tag{96}$$

Each edge $(u,v) \in M \cap Y$, on the other hand, has $W_{\mathscr{E}}(u,\mathscr{E}) + W_{\mathscr{E}}(v,\mathscr{E}) \geq 1$ (see Observation 6.24). Since both these endpoints u,v belong to the node-set $V_Y = V \setminus V_X$, summing over all edges in $M \cap Y$, we get:

$$|M \cap Y| \leq \sum_{v\in V_Y} W_{\mathscr{E}}(v,\mathscr{E}) \tag{97}$$

Since $|M| = |M \cap Y| + |M \setminus Y|$, equations 96 and 98 give us the following upper bound on $|M|$.

$$|M| \leq \sum_{v \in V_Y} W_{\mathscr{E}}(v, \mathscr{E}) + |V_X| \tag{98}$$

To complete the proof, we will now upper bound the size of the set V_X by the sum $\sum_{v \in V_X} W_{\mathscr{E}}(v, \mathscr{E})$. The main idea is simple. Lemma 6.6 shows that only a negligible fraction of the nodes in V_X belong to the set Q_X, and Observation 6.25 shows that for every other node $v \in V_X \setminus Q_X$, the weight $W_{\mathscr{E}}(v, \mathscr{E})$ is very close to one. To be more specific, we infer that:

$$\begin{aligned}
|V_X| &\leq (1 - 32\delta L^4/\varepsilon^2)^{-1} \cdot |V_X \setminus Q_X| && (99)\\
&\leq (1 - 32\delta L^4/\varepsilon^2)^{-1} \cdot (h_1 - \varepsilon - 1/K)^{-1} \cdot \sum_{v \in V_X \setminus Q_X} W_{\mathscr{E}}(v, \mathscr{E}) && (100)\\
&= h_2^{-1} \cdot \sum_{v \in V_X \setminus Q_X} W_{\mathscr{E}}(v, \mathscr{E}) && (101)\\
&\leq h_2^{-1} \cdot \sum_{v \in V_X} W_{\mathscr{E}}(v, \mathscr{E}) && (102)
\end{aligned}$$

Equation 99 follows from Corollary 6.7. Equation 100 follows from Observation 6.25. Equation 101 follows from equation 72.

Since the node-set V is partitioned by the subsets V_X and V_Y, from equations 102 and 98 we infer that:

$$\begin{aligned}
|M| &\leq \sum_{v \in V_Y} W_{\mathscr{E}}(v, \mathscr{E}) + h_2^{-1} \cdot \sum_{v \in V_X} W_{\mathscr{E}}(v, \mathscr{E})\\
&\leq h_2^{-1} \cdot \left(\sum_{v \in V_Y} W_{\mathscr{E}}(v, \mathscr{E}) + \sum_{v \in V_X} W_{\mathscr{E}}(v, \mathscr{E}) \right)\\
&= h_2^{-1} \cdot \sum_{v \in V} W_{\mathscr{E}}(v, \mathscr{E})
\end{aligned}$$

This concludes the proof of the claim. □

As an immediate corollary, we get a bound on $|M|$ in terms of the value of the fractional matching $w_{\mathscr{E}}$.

Corollary 6.27. *We have* $|M| \leq (2/h_2) \cdot \sum_{e \in \mathscr{E}} w_{\mathscr{E}}(e)$.

Proof. Follows from Claim 6.26 and the fact that each multi-edge is incident upon two nodes. □

Step III: Constructing the matching $M^* \subseteq X^* \cup Y$.

Every node in the multigraph $\mathscr{G}_{X^*}$ has degree at most $8Kc$ (see Observation 6.20), and any two given nodes in $\mathscr{G}_{X^*}$ have at most $8K$ multi-edges between them (see Observation 6.21). Hence, by Vizing's theorem [17, 18], there is a proper edge-coloring in $\mathscr{G}_{X^*}$ that uses only $8Kc + 8K = 8K(c+1)$ colors. Let $\lambda \to \{1, \ldots, 8K(c+1)\}$ be such a coloring, where $\lambda(e)$ denotes the color assigned to the multi-edge $e \in \mathscr{E}_{X^*}$. By definition, two multi-edges incident upon the same node get different colors under λ.

We use λ to get a proper edge-coloring $\lambda_{\mathscr{E}} : \mathscr{E} \to \{1, \ldots, 8K(c+1)\}$ of the multi-graph $\mathscr{G}$ with the same number of colors. This is done as follows.

- First, we set $\lambda_{\mathscr{E}}(e) = \lambda(e)$ for every multi-edge $e \in \mathscr{E}_{X^*}$. This ensure that two multi-edges in $\mathscr{E}_{X^*}$ incident upon the same node get different colors under $\lambda_{\mathscr{E}}$, for λ is already a proper coloring of $\mathscr{G}_{X^*}$.

- It remains to assign a color to every multi-edge in $\mathscr{G}_{M\cap Y}$. Towards this end, recall that each multi-edge in $\mathscr{G}_{M\cap Y}$ originates from some edge in $M\cap Y$. Consider any edge $(u,v)\in M\cap Y$. There are two possible cases.
  - *Case 1.* $deg(u,\mathscr{E}_{X^*})+deg(v,\mathscr{E}_{M\cap Y})\geq 8Kc$.
    In this case, we do not have any copy of the edge (u,v) in $\mathscr{G}_{M\cap Y}$. There is nothing to be done as far as the coloring $\lambda_{\mathscr{E}}$ is concerned.
  - *Case 2.* $deg(u,\mathscr{E}_{X^*})+deg(v,\mathscr{E}_{M\cap Y})=8Kc-\mu$, *for some integer* $\mu\geq 1$.
    In this case, there are μ copies of the edge (u,v) in $\mathscr{G}_{M\cap Y}$. While assigning colors to these copies, we only need to ensure that they do not come into conflict with the colors that have already been assigned to the multi-edges in $\mathscr{E}_{X^*}$ incident upon u and v. But, we are guaranteed that there are exactly $(8Kc-\mu)$ of these potentially troubling edges. Since we have a palette of $(8Kc+8K)$ colors to begin with, even after assigning the colors to the edges in $\mathscr{E}_{X^*}$, we are left with at least $(8Kc+8K)-(8Kc-\mu)=8K+\mu$ colors that have not been used on any multi-edge incident upon u or v. Using these leftovers, we can easily color all the μ copies of the edge (u,v) without creating any conflict with the colors assigned to the multi-edges in $\mathscr{E}_{X^*}$.

Thus, there exists a proper coloring of $\mathscr{G}=(V,\mathscr{E})$ using $8K(c+1)$ colors. Since each multi-edge is assigned one color, one of these $8K(c+1)$ colors will hit at least $|\mathscr{E}|/(8K(c+1))$ multi-edges, and those multi-edges will surely constitute a matching (for they cannot share a common endpoint). This shows the existence of a matching $M^*\subseteq X^*\cup Y$ of size at least $|\mathscr{E}|/(8K(c+1))$. Thus, we get the following claim.

Claim 6.28. *There exists a matching $M^*\subseteq X^*\cup Y$ of size at least $|\mathscr{E}|/(8K(c+1))$.*

Step IV: Wrapping things up (The approximation guarantee)

By Claim 6.28, there exists a matching $M^*\subseteq X^*\cup Y$ of size at least $|\mathscr{E}|/(8K(c+1))$. By Claim 6.27, the size of the matching M is at most $(2/h_2)$ times the value of the fractional matching $w_{\mathscr{E}}$ defined on $\mathscr{G}=(V,\mathscr{E})$. Finally, by Observation 6.23, the value of the fractional matching $w_{\mathscr{E}}$ is exactly $|\mathscr{E}|/(8Kc)$. Thus, we conclude that:

$$|M|\leq (2/h_2)\cdot\sum_{e\in\mathscr{E}}w_e(\mathscr{E})=(2/h_2)\cdot\frac{|\mathscr{E}|}{(8Kc)}\leq(2/h_2)\cdot(1+1/c)\cdot|M^*|\leq(2/h_2)\cdot(1+\varepsilon)\cdot|M^*|.$$

The last inequality holds since $c=\lfloor(L^4/2)\rfloor\geq(1/\varepsilon)$ (see Observation 6.16 and equation 54). In other words, there exists a matching $M^*\subseteq X^*\cup Y$ whose size is a $(2/h_2)\cdot(1+\varepsilon)$-approximation to the size of the matching M. This concludes the proof of Lemma 6.8.

7 Maintaining the edge-set of a skeleton: Proof of Theorem 5.7

In this section, we consider the following dynamic setting. We are given an input graph $G=(V,E)$ with $|V|=n$ nodes. In the beginning, the edge-set E is empty. Subsequently, at each time-step, an edge is inserted into (or deleted from) the graph G. However, at each time-step, we know for sure that the maximum degree of any node in G is at most d. Thus, we have the following guarantee.

Lemma 7.1. *We have $deg(v,E)\leq d$ for all nodes $v\in V$.*

We will present an algorithm for maintaining the edge-set X of a skeleton of G (see Definition 5.6).

Roadmap. The rest of this section is organized as follows.

- In Section 7.1, we present a high level overview of our approach.
- In Section 7.2, we define the concepts of a "critical structure" and a "laminar structure" that will be used in later sections. We also motivate the connection between these two concepts and the notion of a skeleton of the graph.
- In Section 7.3, we describe two subroutines that will be crucially used by our algorithm.
- In Section 7.4, we present our algorithm for dynamically maintaining critical and laminar structures and analyze some of its basic properties.
- In Section 7.5, we show that our algorithm in Section 7.4 gives us the edge-set of some skeleton of G. See Theorem 7.26.
- In Section 7.6, we analyze the amortized update time of our algorithm. See Theorem 7.27.
- Finally, Theorem 5.7 follows from Theorem 7.26 and Theorem 7.27.

7.1 A high level overview of our approach

We begin by introducing the concepts of a "critical structure" (see Definition 7.2) and a "laminar structure" (see Definition 7.4). Broadly speaking, in a critical structure we classify each node $v \in V$ as being either "active" or "passive". Barring a few outliers that are called "c-dirty" nodes, every active (resp. passive) node has degree larger (resp. smaller) than $\varepsilon d/L^2$ (resp. $3\varepsilon d/L^2$). While the c-dirty nodes are too few in numbers to have any impact on the approximation ratio of our algorithm, the flexibility of having such nodes turns out to be very helpful if we want to maintain a critical structure in the dynamic setting.

The edge-set $H \subseteq E$ of a critical structure consists of all the edges incident upon active nodes, and a laminar structure consists of a family of $(L_d + 1)$ subsets of these edges $H = H_0 \supseteq H_1 \supseteq \cdots \supseteq H_{L_d}$, where $L_d = \lceil \log_2(d/L^4) \rceil$ (see equation 58). The set H_j is identified as the "layer j" of the laminar structure.

Ignoring some low level details, our goal is to reduce the degree of each active node by a factor of half across successive layers. Thus, the degree of an active node v in the last layer H_{L_d} will be very close to 2^{-L_d} times its degree among the edges in H. Since every edge incident upon an active node is part of H, we get $\deg(v, H_{L_d}) \simeq 2^{-L_d} \cdot \deg(v, H) = 2^{-L_d} \cdot \deg(v, E) = (\lambda_d L^4/d) \cdot \deg(v, E)$ for every active node $v \in V$. The last equality holds since L_d is chosen in such a way that $2^{L_d} = \lambda_d d/L^4$ (see equation 59).

We will also try to ensure that the degree of every node (not necessarily active) reduces by at least a factor of half (it is allowed to reduce by more) across successive layers. This will imply that $\deg(v, H_{L_d})$ is at most $2^{-L_d} \cdot \deg(v, H) = (\lambda_d L^4/d) \cdot \deg(v, H) \leq (\lambda_d L^4/d) \cdot \deg(v, E)$ for every node $v \in V$. Finally, recall that $\deg(v, E) \leq d$ for every node $v \in V$ (see Lemma 7.1), and that $\deg(v, E) < 3\varepsilon d/L^2$ for the passive nodes v. Thus, roughly speaking, we will have $\deg(v, H_{L_d}) \leq \lambda_d L^4$ for all nodes $v \in V$, and $\deg(v, H_{L_d}) \leq 3\varepsilon\lambda_d L^2$ for all passive nodes v.

The main purpose of the preceding discussion was to show the link between the concepts of critical and laminar structures on the one hand, and the notion of a skeleton of a graph on the other. Basically, ignoring the small number of c-dirty nodes, (a) the set of active (resp. passive) nodes correspond to the set of big (resp. tiny) nodes in Definition 5.6, and (b) the edges in the last layer H_{L_d} of the laminar structure correspond to the edge-set X in Definition 5.6. This shows that in order to maintain the edge-set X of a skeleton of G, it suffices to maintain a critical structure (A, P, D_c, H) and a corresponding laminar structure, where the symbols $A, P, D_c \subseteq V$ respectively denote the sets of active, passive and c-dirty nodes.

7.2 Critical and laminar structures

We first define the concept of a "critical structure".

Definition 7.2. *A tuple (A, P, D_c, H), where $A, P, D_c \subseteq V$ and $H \subseteq E$, is called a "critical structure" of $G = (V, E)$ iff the following five conditions are satisfied:*

1. *We have $deg(v, E) > \varepsilon d/L^2$ for all nodes $v \in A \setminus D_c$.*
2. *We have $deg(v, E) < 3\varepsilon d/L^2$ for all nodes $v \in P \setminus D_c$.*

3. *We have:*
$$|D_c| \leq (\delta/(L_d+1)) \cdot |A|. \tag{103}$$
4. *We have* $H = \{(u,v) \in E : \text{ either } u \in A \text{ or } v \in A\}$.
5. *We have* $P = V \setminus A$. *Thus, the node-set* V *is partitioned by the subsets* A *and* P.

The nodes in A *(resp.* P*) are called "active" (resp. "passive"). The nodes in* D_c *(resp.* $V \setminus D_c$*) are called "c-dirty" (resp. "c-clean"), where the symbol "c" stands for the term "critical".*

Intuitively, the above definition says that (a) the active nodes have large degrees in G, (b) the passive nodes have small degrees in G, (c) some nodes (which are called c-dirty) can violate the previous two conditions, but their number is negligibly small compared to the number of active nodes, and (d) the edge-set H consists of all the edges in E with at least one active endpoint. Roughly speaking, the nodes in $A \setminus D_c$ will belong to the set of big nodes B in the skeleton, and the nodes in $P \setminus D_c$ will belong to the set of tiny nodes T (see Definition 5.6). In Section 7.4, we present an algorithm for maintaining a critical structure (A,P,D_c,H). Below, we highlight one important property of a critical structure that follows from Definition 7.2, namely, all the edges in E that are incident upon an active node are part of the set H.

Observation 7.3. *In a critical structure* (A,P,D_c,H)*, we have* $deg(v,H) = deg(v,E)$ *for all* $v \in A$.

We next define the concept of a "laminar structure".

Definition 7.4. *Consider a critical structure* (A,P,D_c,H) *as per Definition 7.2. A laminar structure w.r.t.* (A,P,D_c,H) *consists of* (L_d+1) *"layers"* $0,\ldots,L_d$*. Each layer* $j \in [0,L_d]$ *is associated with a tuple* (C_{lj},D_{lj},H_j) *where* $C_{lj},D_{lj} \subseteq A$ *and* $H_j \subseteq H$*. The nodes in* C_{lj} *(resp.* D_{lj}*) are called* l*-clean (resp.* l*-dirty) at layer* j*, where "l" stands for the term "laminar".*

We will be interested in a laminar structure that satisfies the four invariants described below.

The first invariant states that the edge-sets corresponding to successive layers are contained within one another. In other words, the edge-sets $H_0,\ldots,H_{L_d}$ form a laminar family. Furthermore, we have $H = H_0$.

Invariant 7.5. $H = H_0 \supseteq H_1 \supseteq \cdots \supseteq H_{L_d}$.

The second invariant states that at each layer $j \in [0,L_d]$, each active node is classified as either l-dirty or l-clean. In other words, the set of active nodes is partitioned by the subsets of l-clean and l-dirty nodes at each layer j. Next, just like the edge-sets H_j, the sets of l-clean nodes are also contained within one another across successive layers. The sets $C_0,\ldots,C_{l,L_d}$ also form a laminar family. However, unlike the edge-set H_0, which is always equal to H, the set C_0 can be properly contained within the set of active nodes A. To be more precise, we define $D_{l0} = A \cap D_c$ to be the set of active nodes that are also c-dirty, and $C_{l0} = A \setminus D_{l0}$ to be the set of active nodes that are c-clean.

Invariant 7.6. *The following conditions hold.*
1. $D_{l0} = A \cap D_c$.
2. $D_{lj} \cap C_{lj} = \emptyset$ *and* $D_{lj} \cup C_{lj} = A$ *for every layer* $j \in [0,L_d]$.
3. $D_{l0} \subseteq D_{l1} \subseteq \cdots \subseteq D_{l,L_d}$*, and accordingly,* $A \supseteq C_0 \supseteq C_1 \supseteq \cdots \supseteq C_{l,L_d}$.

Recall the discussion in Section 7.1. In an ideal scenario, we would like to reduce the degree of every active node by a factor of $1/2$ across successive layers. Specifically, we would like to have $\deg(v,H_j) = (1/2) \cdot \deg(v,H_{j-1})$ for every active node $v \in A$ and layer $j \in [1,L_d]$. Such a structure, however, is very difficult to maintain in a dynamic setting. Accordingly, we introduce some slack, and we are happy as long as $\deg(v,H_j)$, instead of being exactly equal to $(1/2) \cdot \deg(v,H_{j-1})$, lies in the range $[1/(2\eta) \cdot \deg(v,H_{j-1}), (\eta/2) \cdot \deg(v,H_{j-1})]$ for $\eta = (1+\gamma/L_d)$. We want every node in the set $C_{lj} \subseteq A$ to satisfy this approximate degree-splitting condition at all the layers $j' \leq j$. In other words, if an active node v is l-clean at layer j, then its degree ought to have been split roughly by half across successive layers in the interval $[0,j]$. This motivates our third invariant.

Invariant 7.7. *For every layer $j \in [1, L_d]$ and every node $v \in C_{lj}$, we have:*

$$(1+\gamma/L_d)^{-1} \cdot \frac{deg(v, H_{j-1})}{2} \leq deg(v, H_j) \leq (1+\gamma/L_d) \cdot \frac{deg(v, H_{j-1})}{2}.$$

At this point, the reader might object that the above invariant only specifies the approximate degree-splitting condition at layer j for the nodes in C_{lj}, whereas our declared goal was to enforce this condition at all the layers in the interval $[0, j]$. Fortunately for us, Invariant 7.6 comes to our rescue, by requiring that a l-clean node at layer j must also l-clean at every layer $j' \leq j$. Hence, Invariants 7.6 and 7.7 together imply the desired guarantee, namely, that the degree of a node $v \in C_{lj}$ is split approximately by half across successive layers in the entire range $[0, j]$. The next lemma states this simple observation in a formal language.

Lemma 7.8. *A laminar structure has for each layer $j \in [1, L_d]$ and each node $v \in C_{lj}$:*

$$\frac{deg(v, E)}{2^j \cdot (1+\gamma/L_d)^j} \leq deg(v, H_j) \leq (1+\gamma/L_d)^j \cdot \frac{deg(v, E)}{2^j}.$$

Proof. Consider any layer $j \in [1, L_d]$ and any node $v \in C_{lj}$. Since $C_{lj} \subseteq C_{lk}$ for all $k \in [1, j]$ (see Invariant 7.6), we have $v \in C_{lk}$ for every layer $k \in [1, j]$. Hence, by Invariant 7.7 we have:

$$\frac{deg(v, H_{k-1})}{2 \cdot (1+\gamma/L_d)} \leq \deg(v, H_k) \leq (1+\gamma/L_d) \cdot \frac{\deg(v, H_{k-1})}{2} \quad \text{for all } k \in [1, j]. \tag{104}$$

From equation 104 we infer that:

$$\frac{deg(v, H_0)}{2^j \cdot (1+\gamma/L_d)^j} \leq \deg(v, H_j) \leq (1+\gamma/L_d)^j \cdot \frac{\deg(v, H_0)}{2^j}. \tag{105}$$

Since $C_{lj} \subseteq A$ (see Invariant 7.6) and $v \in C_{lj}$, we have $v \in A$. Hence, Observation 7.3 and Invariant 7.5 imply that $\deg(v, H_0) = \deg(v, H) = \deg(v, E)$. The lemma now follows from equation 105. □

We now bound the degree of a l-clean node in the last layer L_d in terms of its degree in the graph G.

Corollary 7.9. *For each node $v \in C_{l,L_d}$ in a laminar structure, we have:*

$$e^{-\gamma} \cdot (\lambda_d L^4/d) \cdot deg(v, E) \leq deg(v, H_{L_d}) \leq e^{\gamma} \cdot (\lambda_d L^4/d) \cdot deg(v, E).$$

Proof. Fix any node $v \in C_{l,L_d}$. Setting $j = L_d$ in Lemma 7.8, we get:

$$\frac{deg(v, E)}{2^j \cdot (1+\gamma/L_d)^{L_d}} \leq \deg(v, H_j) \leq (1+\gamma/L_d)^{L_d} \cdot \frac{\deg(v, E)}{2^j}.$$

The corollary now follows from equation 59 and the fact that $(1+\gamma/L_d)^{L_d} \leq e^{\gamma}$. □

We want to ensure that the edge-set H_{L_d} corresponds to the edge-set X of the skeleton in Definition 5.6. Comparing the guarantee stated in Corollary 7.9 with the condition (5) in Definition 5.6, it becomes obvious that each l-clean node in the last layer L_d will belong to the set of big, non-spurious nodes $B \setminus S$ in a skeleton. Furthermore, pointing towards the similarities between the conditions (1), (2) in Definition 7.2 and the conditions (2), (3) in Definition 5.6, we hope to convince the reader that every active, c-clean node will belong to the set B, and that every passive, c-clean node will belong to the set T. This, however, is not the end of the story, for the condition (1) in Definition 5.6 requires that the sets B and T actually partition the set of all nodes V, whereas we have not yet assigned the c-dirty nodes to either B or T. There is an easy fix

for this. We will simply assign all the c-dirty nodes with $\deg(v,E) > \varepsilon d/L^2$ to B, and the remaining c-dirty nodes to T. This will take care of the conditions (1), (2), (3) in Definition 5.6.

By Invariant 7.6, we have $A \setminus D_c = A \setminus D_{l0} = C_{l0}$. Hence, at this point the reader might raise the following objection: So far we have argued that the nodes in $A \setminus D_c = C_{l0}$ will belong to the set B and that the nodes in C_{l,L_d} will belong to the set $B \setminus S$. Intuitively, this alludes to the fact that the nodes in $C_{l0} \setminus C_{l,L_d} = D_{l,L_d} \setminus D_{l0}$ will belong to the set S of spurious nodes. As we will see in Section 7.5, this is indeed going to be the case. But the condition (4) in Definition 5.6 places an upper bound on the maximum number of spurious nodes we can have.[6] Till now, we have not stated any such analogous upper bound on the maximum number of nodes in D_{l,L_d}. To alleviate this concern, we introduce our fourth and final invariant.

Invariant 7.10. *We have:*

$$|D_{lj}| \leq \frac{(j+1)\delta}{(L_d+1)} \cdot |A| \quad \textit{for every layer } j \in [0, L_d].$$

Note that the invariant places an upper bound on the number of l-dirty nodes *at every layer* $j \in [0, L_d]$. At first glance, this might seem to be too stringent a requirement, for all that we initially asked for was an upper bound on the number of l-dirty nodes in layer L_d. However, as we shall see in later sections, this invariant will be very helpful in bounding the amortized update time of our algorithm for maintaining critical and laminar structures.

The remaining two conditions (6) and (7) in Definition 5.6 *cannot* be derived from the Definitions 7.2, 7.4 and the four invariants stated above. Instead, they will follow from some specific properties of our algorithm for maintaining the critical and laminar structures. Ignoring some low level details, these properties are:

1. Most of the time during the course of our algorithm, the layers in the range $[1, L_d]$ keep losing edges (i.e., no edge is inserted into H_j for $j \geq 1$) and hence the degrees of the nodes in these layers keep decreasing. We emphasize that this property is not true for $j = 0$.
2. Consider a time-instant t where we see an exception to the rule (1), i.e., an edge is inserted into some layer $j \in [1, L_d]$ by our algorithm. Then there exists a layer $j' \in [1, j-1]$ with the following property: Just after the time-instant t, the degree of every node drops roughly by a factor of 2 across successive layers in the interval $[j', L_d]$.
3. Using (1) and (2), we upper bound the maximum degree a node can have in any layer $j \in [1, L_d]$. As a corollary, we can show that $\deg(v, H_{L_d}) \leq \lambda_d L^4 + 2$ for all nodes $v \in V$, and that $\deg(v, H_{L_d}) \leq 3\varepsilon\lambda_d L^2 + 2$ for all nodes $v \in P \setminus D_c$. As we will see later on, these two bounds will respectively correspond to the conditions (6) and (7) in Definition 5.6.

To conclude this section, we note that these crucial properties of our algorithm are formally derived and stated in Section 7.4.4 (see Lemmas 7.22, 7.23 and Corollary 7.24).

7.3 Two basic subroutines

In this section, we describe two subroutines that will be heavily used by our algorithm in Section 7.4.

SPLIT($\mathscr{E}$). The first subroutine is called SPLIT($\mathscr{E}$) (see Figure 1). This takes as input an edge-set $\mathscr{E}$ defined over the nodes in V. Roughly speaking, its goal is to reduce the degree of every node by a factor of half, and it succeeds in its goal for all the nodes with degree at least three. To be more precise, the output of the subroutine is a subset of edges $\mathscr{E}' \subseteq \mathscr{E}$ that satisfies the two properties stated in Figure 1. The subroutine runs in $O(|\mathscr{E}|)$ time, and can be implemented as follows:

- Create a new node v^*. For every node $v \in V$ with odd degree $\deg(v, \mathscr{E})$, create a new edge (v^*, v). Let $\mathscr{E}_{new}$ denote the set of newly created edges, and let $V^* = V \cup \{v^*\}$ and $\mathscr{E}^* = \mathscr{E} \cup \mathscr{E}_{new}$. It is easy to

[6] The reader might point out that the condition (3) in Definition 7.2 places an upper bound on the number of c-dirty nodes. However, it is easy to check that the size of the set D_{l,L_d} can potentially be much larger than that of D_c.

check that every node in the graph $G^* = (V, \mathscr{E}^*)$ has an even degree. Hence, in $O(|\mathscr{E}^*|)$ time, we can compute an Euler tour in G^*. We construct the edge-set $\mathscr{E}'$ by first selecting the alternate edges in this Euler tour, and then discarding those selected edges that are part of $\mathscr{E}_{new}$. The subroutine returns the edge-set $\mathscr{E}'$ and runs in $O(|\mathscr{E}^*|) = O(|\mathscr{E}|)$ time. See the paper [9] for details.

1. The input is a set of edges $\mathscr{E}$ defined over the nodes in V.
2. The output is a subset $\mathscr{E}' \subseteq \mathscr{E}$, with the following property:
   For every node $v \in \mathscr{V}$, we have $\frac{\deg(v,\mathscr{E})}{2} - 1 \leq \deg(v,\mathscr{E}') \leq \frac{\deg(v,\mathscr{E})}{2} + 1$.
3. The subroutine runs in $O(|\mathscr{E}|)$ time.

Figure 1: SPLIT($\mathscr{E}$).

REBUILD(j). The second subroutine is called REBUILD(j), where $j \in [1, L_d]$. We will ensure that during the course of our algorithm, the conditions stated in Figure 2 are satisfied just before every call to REBUILD(j). As the name suggests, the subroutine REBUILD(j) will then rebuild the layers $k \in [j, L_d]$ from scratch. See Figure 3 for details. We will show that at the end of the call to REBUILD(j), the Invariants 7.5, 7.6, 7.7 will continue to hold, and Invariant 7.10 will hold for all layers $j' \in [0, L_d]$. We will also derive some nice properties of this subroutine that will be useful later on.

1. The tuple (A, P, D_c, H) is a critical structure as per Definition 7.2.
2. The corresponding laminar structure (see Definition 7.4) satisfies Invariants 7.5, 7.6 and 7.7.
3. Invariant 7.10 holds at every layer $j' \in [0, j-1]$.
4. Invariant 7.10 is violated at layer j.

Figure 2: Initial conditions just before a call to REBUILD(j), $j \in [1, L_d]$.

1. FOR $k = j$ to L_d
2. $\quad H_k \leftarrow \text{SPLIT}(H_{k-1})$.
3. $\quad D_{l,k} \leftarrow D_{l,k-1}$.
4. $\quad C_{l,k} \leftarrow C_{l,k-1}$.

Figure 3: REBUILD(j), $j \in [1, L_d]$. Just before a call to this subroutine, the conditions in Figure 2 hold.

The subroutine SPLIT($\mathscr{E}$) outputs a subset of edges $\mathscr{E}' \subseteq \mathscr{E}$ where the degree of each node is half times its original degree, plus-minus one (see Figure 1). Since the subroutine REBUILD(j) iteratively sets $H_k \leftarrow \text{SPLIT}(H_{k-1})$ for $k = j$ to L_d (see Figure 3), we can get a nearly tight bound on the degree of a node in a layer $k \in [j, L_d]$ when the subroutine REBUILD(j) finishes execution.

Lemma 7.11. *Fix any $j \in [1, L_d]$. At the end of a call to the subroutine REBUILD(j), we have:*

$$\frac{deg(u, H_{k-1})}{2} - 1 \leq deg(u, H_k) \leq \frac{deg(u, H_{k-1})}{2} + 1 \quad \text{for all } u \in V, k \in [j, L_d].$$

The next lemma follows directly from the descriptions in Figures 2 and 3.

Lemma 7.12. *Consider any call to the subroutine REBUILD(j) during the course of our algorithm.*

- *The call does not alter the sets A, P, D_c and H. By Figure 2, the tuple (A, P, D_c, H) is a critical structure (see Definition 7.2) just before the call, and it continues to remain so at the end of the call.*
- *The call does not alter the layers $k < j$, i.e., the sets $\{D_{lk}, C_{lk}, H_k\}, k \in [0, j-1]$, do not change.*
- *Finally, at the end of the call we have: $D_{lk} = D_{l,j-1}$ and $C_{lk} = C_{l,j-1}$ for all layers $k \in [j, L_d]$.*

The proof of the next lemma appears in Section 7.3.1.

Lemma 7.13. *At the end of a call to REBUILD(j), Invariants 7.5, 7.6, 7.7 and 7.10 are satisfied.*

7.3.1 Proof of Lemma 7.13

Throughout this section, we fix a layer $j \in [1, L_d]$ and a given call to the subroutine REBUILD(j). We let t_{start} denote the point in time just before the call to REBUILD(j), whereas we let t_{end} denote the point in time just after the call to REBUILD(j). We consider all the four invariants one after another.

Proof for Invariant 7.5.

By Figure 2, we have $H = H_0 \supseteq H_1 \supseteq \cdots \supseteq H_{j-1}$ at time t_{start}. The call to REBUILD(j) does not change any of the sets $H, H_0, \ldots, H_{j-1}$, and it iteratively sets $H_k \leftarrow \text{SPLIT}(H_{k-1})$ for $k = j$ to L_d. Hence, by Figure 1, we have $H_{j-1} \supseteq H_j \supseteq \cdots \supseteq H_{L_d}$ at time t_{end}. Putting all these observations together, we have $H = H_0 \supseteq H_1 \supseteq \cdots \supseteq H_{L_d}$ at time t_{end}. This shows that Invariant 7.5 is satisfied at time t_{end}.

Proof for Invariant 7.6.

By Figure 2, at time t_{start} we have:

- $D_{l0} = D_c \cap A$.
- $D_{lk} \cap C_{lk} = \emptyset$ and $D_{lk} \cup C_{lk} = A$ for all layers $k \in [1, j-1]$.
- $D_{l0} \subseteq D_{l1} \subseteq \cdots \subseteq D_{l,j-1}$ and $A \supseteq C_{l0} \supseteq C_{l1} \supseteq \cdots \supseteq C_{l,j-1}$.

The call to REBUILD(j) does not change the sets D_c and A. The sets $\{D_{lk}, C_{lk}\}, k < j$, are also left untouched. Finally, it is guaranteed that at time t_{end} we have:

- $D_{lk} = D_{l,j-1}$ and $C_{lk} = C_{l,k-1}$ for all layers $k \in [j, L_d]$.

From all these observations, we infer that Invariant 7.6 holds at time t_{end}.

Proof for Invariant 7.10.

By Figure 2 (item 3), at time t_{start} we have:

$$|D_{lk}| \leq \frac{\delta(k+1)}{(L_d+1)} \cdot |A| \quad \text{at every layer } k \in [0, j-1].$$

The call to REBUILD(j) does not change the set A. Neither does it change the sets $\{D_{lk}\}, k < j$. It is also guaranteed that at time t_{end} we have $D_{lk} = D_{l,j-1}$ for all layers $k \in [j, L_d]$. Thus, at time t_{end} we have:

$$|D_{lk}| = |D_{l,j-1}| \leq \frac{\delta j}{(L_d+1)} \cdot |A| < \frac{\delta(k+1)}{(L_d+1)} \cdot |A| \quad \text{at every layer } k \in [j, L_d].$$

From all these observations, we infer that Invariant 7.10 holds at time t_{end}.

Proof for Invariant 7.7.

Invariant 7.7 holds at time t_{start}, and a call to REBUILD(j) does not alter the layers $k < j$. Thus, we have:

$$\text{Invariant 7.7 holds at time } t_{\text{end}} \text{ for every layer } k \in [1, j-1] \text{ and node } u \in C_{l,k}. \tag{106}$$

Accordingly, we only need to focus on the layers $k \in [j, L_d]$. Since the set $C_{l,j-1}$ does not change during the call to REBUILD(j), we will refer to $C_{l,j-1}$ without any ambiguity. Further, the call to REBUILD(j) ensures that $C_{l,k} = C_{l,j-1}$ for all $k \in [j, L_d]$ at time t_{end}. Thus, in order to prove that Invariant 7.7 holds at time t_{end} in the remaining layers $k \in [j, L_d]$, it suffices to show Claim 7.14.

Claim 7.14. *Consider any node $v \in C_{l,j-1}$. At time t_{end}, we have:*

$$(1+\gamma/L)^{-1} \cdot \frac{deg(v, H_{k-1})}{2} \leq deg(v, H_k) \leq (1+\gamma/L) \cdot \frac{deg(v, H_{k-1})}{2} \quad \text{at every layer } k \in [j, L_d].$$

Throughout the rest of this section, we fix a node $v \in C_{l,j-1}$, and focus on proving Claim 7.14 for the node v. The main idea is very simple: The call to REBUILD(j) ensures that upon its return, $\deg(v, H_k)$ is very close to $(1/2) \cdot \deg(v, H_{k-1})$ for all layers $k \in [j, L_d]$ (see Lemma 7.11). Barring some technical details, this is sufficient to show that $\deg(v, H_k)$ is within the prescribed range at each layer $k \in [j, L_d]$.

Claim 7.15. *At time t_{end}, we have:*

$$\frac{deg(v, H_{k-1})}{2} - 1 \leq deg(v, H_k) \leq \frac{deg(v, H_{k-1})}{2} + 1 \quad \text{at every layer } k \in [j, L_d].$$

Proof. Follows from Lemma 7.11. □

Claim 7.16. *At time t_{end}, we have $deg(v, H_k) \geq L$ at every layer $k \in [j-1, L_d]$.*

Proof. □

Claim 7.17. *Consider any number $x \geq L$. We have:*

- $x/2 - 1 \geq (x/2) \cdot (1+\gamma/L_d)^{-1}$, *and*
- $x/2 + 1 \leq (x/2) \cdot (1+\gamma/L_d)$.

Proof. To prove the first part of the claim, we infer that:

$$\begin{aligned}
(x/2) - (x/2) \cdot (1+\gamma/L_d)^{-1} &= \left(\frac{x}{2}\right) \cdot \left(\frac{\gamma/L_d}{1+\gamma/L_d}\right) \\
&\geq \left(\frac{L\gamma}{2L_d}\right) \cdot \left(\frac{1}{1+\gamma/L_d}\right) \\
&\geq \frac{L\gamma}{4L_d} \qquad (107) \\
&\geq 1 \qquad (108)
\end{aligned}$$

Equation 107 follows from equation 66. Equation 108 follows from equation 67.

To prove the second part of the claim, we infer that:

$$\begin{aligned}
(x/2) \cdot (1+\gamma/L_d) - (x/2) &= (x/2) \cdot (\gamma/L_d) \\
&\geq \frac{L\gamma}{2L_d} \\
&\geq 1 \qquad (109)
\end{aligned}$$

Equation 109 follows from equation 67. □

Claim 7.18. *At time t_{end}, we have*

$$(1+\gamma/L)^{-1}\cdot\frac{deg(v,H_{k-1})}{2}\leq deg(v,H_k)\leq(1+\gamma/L)\cdot\frac{deg(v,H_{k-1})}{2}\quad \text{at every layer } k\in[j,L_d].$$

Proof. Consider any layer $k\in[j,L_d]$. Let x be the value of $\deg(v,H_{k-1})$ at time t_{end}. Hence, at time t_{end}, the value of $\deg(v,H_k)$ lies in the range $[x/2-1,x/2+1]$ (see Claim 7.15). Since $x\geq L$ (see Claim 7.16), we infer that the range $[x/2-1,x/2+1]$ is completely contained within the range $\left[(1+\gamma/L)^{-1}\cdot(x/2),(1+\gamma/L)\cdot(x/2)\right]$ (see Claim 7.17). Thus, at time t_{end}, the value of $\deg(v,H_k)$ also falls within the range $\left[(1+\gamma/L)^{-1}\cdot(x/2),(1+\gamma/L)\cdot(x/2)\right]$. This concludes the proof of the claim. □

Claim 7.14 follows from Claim 7.18.

7.4 Our algorithm for maintaining critical and laminar structures

We use the term "edge-update" to refer to the insertion/deletion of an edge in the graph $G=(V,E)$. Thus, in a dynamic setting the graph $G=(V,E)$ changes due to a sequence of edge-updates. In this section, we present an algorithm that maintains a critical structure and a laminar structure in such a dynamic setting, and ensures that all the invariants from Section 7.2 are satisfied. Before delving into technical details, we first present a high level overview of our approach.

A brief outline of our algorithm. Our algorithm works in "phases", where the term "phase" refers to a contiguous block of edge-updates. In the beginning of a phase, there are no dirty nodes, i.e., we have $D_c=\emptyset$ and $D_{lj}=\emptyset$ for all $j\in[0,L_d]$, and all the four invariants from Section 7.2 are satisfied.

The algorithm in the middle of a phase. We next describe how to modify the critical structure after an edge-update in the middle of a phase. Towards this end, consider an edge-update that corresponds to the insertion/deletion of the edge (u,v) in $G=(V,E)$. This edge-update changes the degrees of the endpoints u,v, and this might lead to some node $x\in\{u,v\}$ violating one of the first two conditions in Definition 7.2. But this can happen only if the node x was c-clean before the edge-update. Hence, the easiest way to fix the problem, if there is one, is to change the status of the node to c-dirty. And we do exactly the same. Next, we ensure that H remains the set of edges incident upon the active nodes. Towards this end, we check if either of the endpoints u,v is active. If the answer is yes, then the edge (u,v) is inserted into (resp. deleted from) H along with the insertion (resp. deletion) of (u,v) in $G=(V,E)$.

From the above discussion, it is obvious that the sets of active and passive nodes do not change in the middle of a phase. And we ensure that H remains the set of edges incident upon the active nodes. In contrast, as the phase goes on, we see more and more c-clean nodes becoming c-dirty. The set of c-dirty nodes, accordingly, keeps getting larger along with the passage of time. The phase terminates when the size of the set D_c exceeds the threshold $(\delta/(L_d+1))\cdot|A|$, thereby violating the condition (3) in Definition 7.2.

We next show how to maintain the laminar structure in the middle of a phase.

1. Whenever an edge (u,v) is inserted into H, we set $H_0\leftarrow H_0\cup\{(u,v)\}$, and whenever an edge (u,v) is deleted from the graph, we set $H_j\leftarrow H_j\setminus\{(u,v)\}$ for all layers $j\in[0,L_d]$. This ensures that we always have $H=H_0\supseteq H_1\supseteq\cdots\supseteq H_{L_d}$. Furthermore, whenever an active node v becomes c-dirty, we set $D_{lj}\leftarrow D_{lj}\cup\{v\}$ for all layers $j\in[0,L_d]$. This ensures that Invariant 7.6 is satisfied.
2. Whenever we see a node $v\in C_{l,j}$ violating Invariant 7.7 at layer $j\in[1,L_d]$, we set $D_{l,k}\leftarrow D_{l,k}\cup\{v\}$ and $C_{l,k}\leftarrow C_{l,k}\setminus\{v\}$ for all layers $k\in[j,L_d]$. This restores the validity of Invariant 7.7 without tampering with Invariant 7.6.
3. Whenever Invariant 7.10 gets violated, we find the smallest index j at which $|D_{lj}|>(\delta(j+1)/(L_d+1))\cdot|A|$, and call the subroutine REBUILD(j) (see Figure 3).

Fix any layer $j\in[1,L_d]$. A call to REBUILD(j') with $j'>j$ does not affect the layers in the range $[0,j]$. Consequently, the set D_{lj} (resp. C_{lj}) keeps getting bigger (resp. smaller) till the point arrives where

REBUILD(j') is called for some $j' \leq j$.[7] To be more specific, in the middle of a phase, a node v can switch from D_{lj} to C_{lj} only if REBUILD(j') is called for some $j' \leq j$. In a similar vein, an edge gets inserted into H_j only if REBUILD(j') is called for some $j' \leq j$; and at other times in the middle of a phase the set H_j can only keep shrinking (see item (1) above). To summarize, the reader should note that the algorithm satisfies some nice monotonicity properties. These will be very helpful in our analysis in later sections.

Dealing with the termination of a phase. When a phase terminates, we need to do some cleanup work before starting the next phase. Specifically, recall that a phase terminates when the number of c-dirty nodes goes beyond the acceptable threshold. At this stage, we shift some active c-dirty nodes $v \in A \cap D_c$ from the set A to the set P, shift some passive c-dirty nodes $v \in P \cap D_c$ from the set P to the set A, and finally set $D_c = \emptyset$. At the end of these operations, we have $D_c = \emptyset$, and we perform these operations in such a way that Definition 7.2 is satisfied. Next, we construct the entire laminar structure from scratch by calling the subroutine REBUILD(1) (see Figure 3). Subsequently, we start the next phase.

Roadmap. We will now present the algorithm in details. The rest of this section is organized as follows.

- In Section 7.4.1, we state the initial conditions that hold in the beginning of a phase.
- In Section 7.4.2, we describe our algorithm in the middle of a phase.
- In Section 7.4.3, we describe the cleanup that needs to be performed at the end of a phase.
- In Section 7.4.4, we derive some useful properties of our algorithm.

7.4.1 Initial conditions in the beginning of a phase

Just before the first edge insertion/deletion of a phase, the following conditions are satisfied.

1. There are no c-dirty nodes, i.e., $D_c = \emptyset$.
2. There are no l-dirty nodes at any layer $j \in [0, L_d]$, i.e., $D_{lj} = \emptyset$ for all $j \in [0, L_d]$.

In the beginning of the very first phase, the graph $G = (V, E)$ has an empty edge-set. At that instant, every node is passive and the conditions (1) and (2) are satisfied.

By induction hypothesis, suppose that the conditions (1) and (2) are satisfied when the k^{th} phase is about to begin, for some integer $k \geq 1$. We will process the edge insertions/deletions in G during the k^{th} phase using the algorithm described in Sections 7.4.2 and 7.4.3. This algorithm will ensure that the conditions (1) and (2) are satisfied at the start of the $(k+1)^{th}$ phase.

7.4.2 Handling edge insertion/deletions in the middle of a phase

Suppose that an edge (u, v) is inserted into (resp. deleted from) the graph $G = (V, E)$ in the middle of a phase. In this section, we will show how to handle this edge-update.

Step I: Updating the critical structure. We update the critical structure (A, P, D_c, H) as follows.

- If the edge (u, v) has at least one active endpoint, i.e., if $\{u, v\} \cap A \neq \emptyset$, then:
  - If we are dealing with the insertion of the edge (u, v) into G, then set $H \leftarrow H \cup \{(u, v)\}$.
    Else if we are dealing with the deletion of the edge (u, v) from G, then set $H \leftarrow H \setminus \{(u, v)\}$.

  This ensures that the condition (4) in Definition 7.2 remain satisfied.

Next, note that the edge-update changes the degrees of the endpoints u, v. Hence, to satisfy the conditions (1) and (2) in Definition 7.2, we perform the following operations on each node $x \in \{u, v\}$.

- If $x \in P \setminus D_c$ and $\deg(x, E) \geq 3\varepsilon d/L^2$, then set $D_c \leftarrow D_c \cup \{x\}$.
  Else if $x \in A \setminus D_c$ and $\deg(x, E) \leq \varepsilon d/L^2$, then set $D_c \leftarrow D_c \cup \{x\}$.

At this point, the conditions (1), (2), (4) and (5) in Definition 7.2 are satisfied. But, we cannot yet be sure about the remaining condition (3). The next step in our algorithm will resolve this issue.

[7] For $j = 0$, we have $D_{l0} = D_c \cap A$. Since the set A remains unchanged and the set D_c keeps getting bigger along with the passage of time in the middle of a phase, we conclude that the same thing happens with the set D_{l0}.

Step II: Deciding if we have to terminate the phase.

Step I of the algorithm might have made changed the status of one or both the endpoints u, v from c-clean to c-dirty. If this is the case, then this increases the number of c-dirty nodes. To find out if this violates the condition (3) in Definition 7.2, we check if $|D_c| > (\delta/(L_d+1)) \cdot |A|$.

- If $|D_c| > (\delta/(d+1)) \cdot |A|$, then the number of c-dirty nodes have increased beyond the acceptable threshold, and so we terminate the current phase and move on to Section 7.4.3. Else if $|D_c| \leq (\delta/(L_d+1)) \cdot |A|$, then all the conditions in Definition 7.2 are satisfied, and we move on to Step III.

Step III: Updating the laminar structure. If we have reached this stage, then we know for sure that the critical structure (A, P, D_c, H) now satisfies all the conditions in Definition 7.2. It only remains to update the laminar structure, which is done as follows.

1. For each $x \in \{u, v\}$:
   - If $x \in A$ and Step I has converted the node x from c-clean to c-dirty, then:
     For all $k \in [0, L_d]$, set $D_{lk} \leftarrow D_{lk} \cup \{x\}$ and $C_{lk} \leftarrow C_{lk} \setminus \{x\}$.

   This ensures that D_{l0} remains equal to $D_c \cap A$ and that Invariant 7.6 continues to hold.
2. If Step I inserts the edge (u, v) into H, then set $H_0 \leftarrow H_0 \cup \{(u, v)\}$.
   Else if Step I deletes the edge (u, v) from H, then set $H_j \leftarrow H_j \setminus \{(u, v)\}$ for all layers $j \in [0, L_d]$.
   This ensures that Invariant 7.5 is satisfied. Also note that if the edge-update under consideration is an insertion, this does not affect the edges in the layers $j \geq 1$.
3. The previous operations might have changed the degree of some endpoint $x \in \{u, v\}$ in the laminar structure, and hence, we have ensure that no node $x \in \{u, v\}$ violates Invariant 7.7. Accordingly, we call the subroutine CLEANUP(x) for each node $x \in \{u, v\}$. See Figure 4. The purpose of these calls is to restore Invariant 7.7 without affecting Invariant 7.6.
   Due to the calls to CLEANUP(u) and CLEANUP(v), the sets D_{lj} get bigger and the set C_{lj} gets smaller. In other words, a node that is l-dirty at some layer j never becomes l-clean at layer j due to these calls. This monotonicity property will be very helpful in the analysis of our algorithm.
4. The calls to CLEANUP(u) and CLEANUP(v) might have increased the number of l-dirty nodes at some layers, and hence, Invariant 7.10 might get violated at some layer $j \in [0, L_d]$.
   Next, note that since we gone past Step II, we must have $|D_c| \leq \delta/(L_d+1) \cdot |A|$. Since $D_{l0} = D_c \cap A$ (see item (1) above), we have $|D_{l0}| \leq |D_c| \leq \delta/(L_d+1) \cdot |A|$. In other words, we only need to be concerned about Invariant 7.10 for layers $j > 0$.
   To address this concern, we now call the subroutine VERIFY(). See Figure 5.
   Lemma 7.19 shows that Invariant 7.10 holds for all layers $j \in [0, L_d]$ at the end of the call to VERIFY().
   At this stage all the invariants hold, and we are ready to handle the next edge-update in G.

```
1.  FOR j = 1 to L_d:
2.      IF x ∈ C_lj and the node x violates Invariant 7.7 at layer j, THEN
3.          Set D_lj' ← D_lj' ∪ {x} and C_lj' ← C_lj' \ {x} for all j' ∈ [j, L_d].
4.          RETURN.
```

Figure 4: CLEANUP(x).

Lemma 7.19. *At the end of a call to VERIFY(), Invariant 7.10 holds for all layers $j \in [0, L_d]$.*

Proof. Since we have gone past Step II, we must have $|D_c| \leq \delta/(L_d+1) \cdot |A|$. Since $D_{l0} = D_c \cap A$ (see item (1) in Step III), we have $|D_{l0}| \leq |D_c| \leq \delta/(L_d+1) \cdot |A|$. In other word, Invariant 7.10 holds for $j = 0$ at the end of the call to VERIFY(). Henceforth, we focus on the layers $j > 0$.

1. FOR $j = 1$ to L_d:
2. IF $|D_{lj}| > (\delta(j+1)/(L_d+1)) \cdot |A|$, THEN
3. Call the subroutine REBUILD(j). See Figure 3.
4. RETURN.

Figure 5: VERIFY().

If the subroutine VERIFY() does not make any call to REBUILD(j') with $j' \in [1, L_d]$ during its execution, then clearly Invariant 7.10 holds for all layers $j \in [1, L_d]$. Otherwise, let $k \in [1, L_d]$ be the index such that the subroutine REBUILD(k) is called during the execution of VERIFY().

Since the subroutine REBUILD(j) was not called for any $j \in [1, k-1]$, it means that Invariant 7.10 was already satisfied for layers $j \in [1, k-1]$. Next, note that the subroutine REVAMP() is terminated after REBUILD(k) finishes execution. Hence, Lemma 7.12 ensures that $C_{lj} = C_{k-1}$ for all $j \in [k, L_d]$ at the end of the call to REVAMP(). So at that instant we have $|D_{lj}| = |D_{l,k-1}|$ for all $j \in [k, L_d]$. Since Invariant 7.10 holds for $j = k-1$, we infer that Invariant 7.10 also holds for all $j \in [k, L_d]$ at that instant. □

7.4.3 Terminating a phase

We terminate the current phase when the number of c-dirty nodes becomes larger than $(\delta/(L_d+1))$ times the number of active nodes. To address this concern, we call the subroutine REVAMP() as described below.

REVAMP().

1. We will first modify the critical structure (A, P, D_c, H). Let $V' \leftarrow D_c$ be the set of c-dirty nodes at the time REVAMP() is called. We run the FOR loop described below.
   FOR ALL nodes $v \in V'$:
   - Set $D_c \leftarrow D_c \setminus \{v\}$. So the node v is no longer c-dirty, and it might violate the conditions (1) and (2) in Definition 7.2. To address this concern, we perform the following operations.
   - If $v \in A$ and $\deg(v, E) < 3\varepsilon d/L^2$, then set $A \leftarrow A \setminus \{v\}$, $P \leftarrow P \cup \{v\}$, and $H \leftarrow H \setminus \{(u,v) \in E : u \in P\}$. In other words, we change the status of the node from active to passive, which satisfies the conditions (1) and (2) in Definition 7.2. Next, to satisfy the condition (4) in Definition 7.2, we delete from the set H those edges whose one endpoint is v and other endpoint is some passive node.
   - Else if $v \in P$ and $\deg(v, E) > \varepsilon d/L^2$, then set $A \leftarrow A \cup \{v\}$, $P \leftarrow P \setminus \{v\}$, and $H \leftarrow H \cup \{(u,v) \in E\}$. In other words, we change the status of the node from active to passive, which satisfies the first and second conditions in Definition 7.2. Next, to satisfy the fourth condition in Definition 7.2, we ensure that all the edges incident upon v belong to H.

   At the end of the For loop, there are no c-dirty nodes, i.e., $D_c = \emptyset$ and all the conditions in Definition 7.2 are satisfied. Thus, at this stage (A, P, D_c, H) is a critical structure with no c-dirty nodes.
2. Next, we update our laminar structure as follows.
   - Set $H_0 \leftarrow H$, and $H_j \leftarrow \emptyset$ for all $j \in [1, L_d]$.
   - Set $D_{l0} \leftarrow \emptyset$ and $D_{lj} \leftarrow A$ for all $j \in [1, L_d]$.
   - Set $C_{l0} \leftarrow A$ and $C_{lj} \leftarrow \emptyset$ for all $j \in [1, L_d]$.

   At this stage, all the conditions stated in Figure 2 are satisfied for $j = 1$. Accordingly, we call the subroutine REBUILD(1). This constructs the layers $j \in [1, L_d]$ of the laminar structure from scratch.

By Lemma 7.13, at the end of the call to REBUILD(1), all the four invariants from Section 7.2 are satisfied. Furthermore, Lemma 7.12 implies that $D_{lj} = D_{l0} = D_c = \emptyset$ for all $j \in [1, L_d]$. Hence, both the conditions (1) and (2) stated in Section 7.4.1 hold at this time. So we start a new phase from the next edge-update.

7.4.4 Some useful properties of our algorithm

In this section we derive some nice properties of our algorithm that will be used in later sections. In Lemma 7.20, we note that our algorithm satisfies all the four invariants from Section 7.2. Next, in Lemma 7.21 (resp. Lemma 7.22), we summarize the way the critical (resp. laminar) structure changes with the passage of time within a given phase. The proofs of these three lemmas follow directly from the description of our algorithm, and are omitted.

Lemma 7.20. *Suppose that we maintain a critical structure and a laminar structure as per our algorithm. Then Invariants 7.5, 7.6, 7.7 and 7.10 are satisfied after each edge-update in the graph $G = (V, E)$.*

Lemma 7.21. *Consider the maintenance of the critical structure in any given phase as per our algorithm.*

- *In the beginning of the phase, there are no c-dirty nodes (i.e., $D_c = \emptyset$).*
- *In the middle of a phase, a node can change from being c-clean to c-dirty, but not the other way round.*
- *The sets of active and passive nodes do not change in the middle of the phase.*
- *The phase ends when the number of c-dirty nodes exceeds the threshold $(\delta/(L_d+1)) \cdot |A|$, and at this point the subroutine REVAMP() is called. At the end of the call to REVAMP(), the next phase begins.*

Lemma 7.22. *Consider the maintenance of the laminar structure in any given phase as per our algorithm.*

- *In the beginning of the phase, there are no l-dirty nodes, i.e., $D_{lj} = \emptyset$ for all $j \in [0, L_d]$.*
- *Consider any layer $j \in [1, L_d]$, and focus on any time interval in the middle of the phase where no call is made to REBUILD(k) with $k \in [1, j]$. During such a time interval:*
  - *No edge gets inserted into the set H_j. In other words, the edge-set H_j keeps shrinking. Furthermore, an edge $e \in H_j$ gets deleted from H_j only if it gets deleted from the graph $G = (V, E)$.*
  - *No node $v \in A$ gets moved from D_{lj} to C_{lj}. In other words, the node-set D_{lj} (resp. C_{lj}) keeps growing (resp. shrinking).*
- *At layer $j = 0$, we have $H_0 = H$ and $D_{l0} = A \cap D_c$. Thus, by Lemma 7.21, the node-set D_{l0} (resp. C_{l0}) keeping growing (resp. shrinking) throughout the duration of the phase.*

Next, we upper bound the maximum degree a node can have in a given layer $j \in [1, L_d]$. We emphasize that the bounds in Lemma 7.23 are artifacts of the specific algorithm we have designed. In other words, these bounds *cannot* be derived only by looking at the definitions and the invariants stated in Section 7.2. Specifically, we exploit two important properties of our algorithm.

- Consider any layer $j \in [1, L_d]$. In a given phase, the degree of a node $v \in V$ in this layer can increase only during a call to REBUILD(k) for some $k \in [1, j]$. See Lemma 7.22.
- Roughly speaking, just after the end of a call to REBUILD(k), the degree of each node $v \in V$ drops by at least a factor of 2 across successive layers in the range $[k, L_d]$. See Lemma 7.11.

The proof of Lemma 7.23 appears in Section 7.4.5.

Lemma 7.23. *Suppose that we maintain a critical structure and a laminar structure as per our algorithm, and let $j \in [0, L_d]$ be any layer in the laminar structure. Then we always have:*

$$deg(v, H_j) \leq d \cdot 2^{-j} + \sum_{j'=0}^{j-1} 2^{-j'} \quad \text{for all nodes } v \in V. \tag{110}$$

$$deg(v, H_j) \leq (3\varepsilon d/L^2) \cdot 2^{-j} + \sum_{j'=0}^{j-1} 2^{-j'} \quad \text{for all nodes } v \in P \setminus D_c. \tag{111}$$

Corollary 7.24. *Suppose that we maintain a critical structure and a laminar structure as per our algorithm. Then we always have:*

1. $deg(v, H_{L_d}) \leq \lambda_d L^4 + 2$ *for all nodes* $v \in V$.

2. $deg(v, H_{L_d}) \leq 3\varepsilon\lambda_d L^2 + 2$ *for all nodes* $v \in P \setminus D_c$.

Proof. Follows from Lemma 7.23 and equation 59. □

7.4.5 Proof of Lemma 7.23

Proof of equation 110.

We prove equation 110 for a given node $v \in V$, using induction on the number of edge-updates seen so far.

- *Base step.* Equation 110 holds for node v after the t^{th} edge-update in G, for $t = 0$.
  The base step is true since initially the graph G is empty and $\deg(v, H_j) = 0$ for all $j \in [0, L_d]$.
- *Induction step.* Suppose that equation 110 holds for node v after the t^{th} edge-update in G, for some integer $t \geq 0$. Given this induction hypothesis, we will show that equation 110 continues to hold for node v after the $(t+1)^{th}$ edge-update in G.

From this point onwards, we focus on proving the induction step. There are two possible cases to consider.

Case 1. The $(t+1)^{th}$ edge-update in G resulted in the termination of a phase.

In this case, the following conditions hold after our algorithm handles the $(t+1)^{th}$ update in G.

$$\deg(v, H_0) \leq d \tag{112}$$

$$\deg(v, H_k) \leq \frac{\deg(v, H_{k-1})}{2} + 1 \quad \text{at every layer } k \in [1, L_d]. \tag{113}$$

Equation 112 holds since d is the maximum degree a node can have in the graph $G = (V, E)$ (see Lemma 7.1) and $H_0 \subseteq E$. Equation 113 holds since the $(t+1)^{th}$ edge-update marks the termination of a phase. Hence, our algorithm calls the subroutine REBUILD(1) while handling the $(t+1)^{th}$ edge-update (see item (2) in Section 7.4.3). At the end of this call to REBUILD(1), the degree of any node in a layer $k \in [1, L_d]$ is $(1/2)$ times its degree in the previous layer, plus-minus one (see Lemma 7.11).

By equations 112 and 113, we get the following guarantee after the $(t+1)^{th}$ edge-update in G.

$$\deg(v, H_j) \leq d \cdot 2^{-j} + \sum_{j'=0}^{j-1} 2^{-j'} \quad \text{at every layer } j \in [0, L_d]. \tag{114}$$

This concludes the proof of the induction step.

Case 2. The $(t+1)^{th}$ edge-update in G does not result in the termination of a phase.

In this case, the $(t+1)^{th}$ edge-update falls in the middle of a phase, and is handled by the algorithm in Section 7.4.2. Specifically, the edge-sets $H_0, \ldots, H_{L_d}$ are modified by the procedure described in Step III (see Section 7.4.2). There are two operations performed by this procedure that concern us, for they are the only ones that tamper with the edge-sets $H_0, \ldots, H_{L_d}$.

- (a) In item (2) of Step III (Section 7.4.2), we might change some of the edge-sets $H_0, \ldots, H_{L_d}$.
- (b) In item (4) of Step III (Section 7.4.2) we call VERIFY(), which in turn might call REBUILD(j) for some layer $j \in [1, L_d]$. And a call to REBUILD(j) reconstructs the edge-sets $H_j, \ldots, H_{L_d}$.

Operation (a) never inserts an edge into a layer $j > 0$. Thus, due to this operation $\deg(v, H_j)$ can only decrease, provided $j > 0$. This ensures that for each layer $j > 0$, $\deg(v, H_j)$ continues to satisfy the desired upper bound of equation 110 after operation (a). The value of $\deg(v, H_0)$, however, can increase due to operation (a). But this does not concern us, for we always have $\deg(v, H_0) \leq \deg(v, E) \leq d$ (see Lemma 7.1). So the upper bound prescribed by equation 110 for layer $j = 0$ is trivially satisfied all the time.

Operation (b) tampers with the edge-sets $H_0, \ldots, H_{L_d}$ only if the subroutine REBUILD(j), for some $j \in [1, L_d]$, is called during the execution of VERIFY(). We focus on this call to REBUILD(j). Just before

the call begins, equation 110 holds for node v. The call to REBUILD(j) does not alter the layers $j' \in [0, j-1]$ (see Lemma 7.12). Accordingly, at the end of the call to REBUILD(j), we have:

$$\deg(v, H_k) \leq d \cdot 2^{-k} + \sum_{j'=0}^{k-1} 2^{-j'} \quad \text{at every layer } k \in [0, j-1]. \tag{115}$$

Furthermore, the call to REBUILD(j) ensures that the degree of a node at any layer $k \in [j, L_d]$ is half of its degree in the previous layer, plus-minus one (see Lemma 7.11). Accordingly, at the end of the call to REBUILD(j), we have:

$$\deg(v, H_k) \leq \frac{\deg(v, H_{k-1})}{2} + 1 \quad \text{at every layer } k \in [j, L_d]. \tag{116}$$

By equations 115 and 116, at the end of the call to REBUILD(j) we have:

$$\deg(v, H_k) \leq d \cdot 2^{-k} + \sum_{j'=0}^{k-1} 2^{-j'} \quad \text{at every layer } k \in [j, L_d]. \tag{117}$$

Equations 115 and 117 conclude the proof of the induction step.

Proof of equation 111.

Throughout the proof, fix any node $v \in V$. We will show that the node satisfies equation 111 in every "phase" (see Section 7.4). Accordingly, fix any phase throughout the rest of the proof. Also recall Lemma 7.21, which summarizes the way the critical structure changes along with the passage of time within a given phase. Since the node-set V is partitioned into the subsets A and P, there are two cases to consider.

- *Case 1.* The node v is active (i.e., part of the set A) in the beginning of the phase. Here, the node continues to remain active throughout the duration of the phase. So the lemma is trivially true for v.
- *Case 2.* The node v is passive (i.e., part of the set P) in the beginning of the phase. Here, the node continues to remain passive till the end of the phase. We will consider two possible sub-cases.
  - *Case 2a. The node never becomes c-dirty during the phase.* In this case, we have $v \in P \setminus D_c$ and $\deg(v, E) \leq (3\varepsilon d / L^2)$ throughout the duration of the phase (see Definition 7.2). Hence, we can adapt the proof of equation 110 by replacing d with $(3\varepsilon d / L^2)$ as an upper bound on the maximum degree of the node, and get the following guarantee: Throughout the duration of the phase, we have $\deg(v, H_j) \leq (3\varepsilon d L^{-2})/2^j + \sum_{j'=0}^{j-1} 1/2^{j'}$ for every layer $j \in [0, L_d]$.
  - *Case 2b. The node becomes c-dirty at some time t in the middle of the phase, and from that point onwards, the node remains c-dirty till the end of the phase.* In this case, the lemma is trivially true for the node from time t till the end of the phase. Hence, we consider the remaining time interval from the beginning of the phase till time t. During this interval, we have $v \in P \setminus D_c$, and we can apply the same argument as in Case 2a. We thus conclude that the lemma holds for the node v throughout the duration of the phase.

7.5 Maintaining the edge-set of an skeleton

In Section 7.2, we defined the concepts of a critical structure and a laminar structure in the graph $G = (V, E)$. In Section 7.4, we described our algorithm for maintaing these two structures in a dynamic setting. In this section, we will show that the edges $e \in H_{L_d}$ (i.e., the edges from the last layer of the laminar structure) constitute the edge-set X of a skeleton of the graph G (see Definition 5.6). Towards this end, we define the

node-sets $B, T, S \subseteq V$ and the edge-set $X \subseteq E$ as follows.

$$S = D_c \cup D_{l,L_d} \tag{118}$$
$$B = (A \setminus S) \cup \left\{ v \in S : \deg(v,E) > \varepsilon d/L^2 \right\} \tag{119}$$
$$T = (P \setminus S) \cup \left\{ v \in S : \deg(v,E \leq \varepsilon d/L^2 \right\} \tag{120}$$
$$X = H_{L_d} \tag{121}$$

The next lemma shows that the node-sets $B, T, S \subseteq V$ and the edge-set $X \subseteq E$ as defined above satisfies all the seven properties stated in Definition 5.6. This lemma implies the main result of this section, which is summarized in Theorem 7.26. As far as the proof of Lemma 7.25 is concerned, the reader should note that the conditions (1) – (5) follow directly from Definitions 7.2, 7.4 and the four invariants stated in Section 7.2. In other words, all the critical and laminar structures that satisfy these invariants will also satisfy the conditions (1) – (5). In sharp contrast, we need to use a couple of crucial properties of our algorithm (see Corollaries 7.24 and 7.24) to prove the remaining two conditions (6) and (7). And it is not difficult to see that there are instances of critical and laminar structures that satisfy the four invariants from Section 7.2 but do not satisfy the conditions (6) and (7).

Lemma 7.25. *Suppose that a critical structure (A, P, D_c, H) and a laminar structure $(H_1, \ldots, H_{L_d})$ are maintained as per the algorithm in Section 7.4. Also suppose that the node-sets $B, T, S \subseteq V$ and the edge-set $X \subseteq E$ are defined as in equations 118, 119, 120 and 121. Then the following conditions hold.*

1. *We have $B \cup T = V$ and $B \cap T = \emptyset$.*
2. *For every node $v \in B$, we have $deg(v,E) > \varepsilon d/L^2$.*
3. *For every node $v \in T$, we have $deg(v,E) < 3\varepsilon d/L^2$.*
4. *We have $|S| \leq 4\delta \cdot |B|$.*
5. *For every node $v \in B \setminus S$, we have:*

$$e^{-\gamma} \cdot (\lambda_d L^4/d) \cdot deg(v,E) \leq deg(v,X) \leq e^{\gamma} \cdot (\lambda_d L^4/d) \cdot deg(v,E).$$

6. *For every node $v \in V$, we have $deg(v,X) \leq \lambda_d L^4 + 2$.*
7. *For every node $v \in T \setminus S$, we have $deg(v,X) \leq 3\varepsilon\lambda_d L^2 + 2$.*

Proof.

1. Recall that $A \subseteq V$ and $P = V \setminus A$ (see Definition 7.2). Since $S \subseteq V$ (see equation 118 and Definitions 7.2, 7.4), we infer that the node-set V is partitioned into three subsets: $A \setminus S$, $P \setminus S$ and S. Hence, from equations 119 and 120, we get: $B \cup T = V$ and $B \cap T = \emptyset$.
2. Definition 7.2 implies that $\deg(v,E) > \varepsilon d/L^2$ for all nodes $v \in A \setminus D_c$. Since $D_c \subseteq S$ (see equation 118), we get: $\deg(v,E) > \varepsilon d/L^2$ for all nodes $v \in A \setminus S$. Hence, equation 119 implies that $\deg(v,E) > \varepsilon d/L^2$ for all nodes $v \in B$.
3. Definition 7.2 implies that $\deg(v,E) < 3\varepsilon d/L^2$ for all nodes $v \in P \setminus D_c$. Since $D_c \subseteq S$ (see equation 118), we get: $\deg(v,E) < 3\varepsilon d/L^2$ for all nodes $v \in P \setminus S$. Hence, equation 120 implies that $\deg(v,E) < 3\varepsilon d/L^2$ for all nodes $v \in T$.
4. The basic idea behind the proof of condition (4) is as follows. Equation 103 implies that $|D_c| \leq \delta \cdot |A|$. Since $S = D_c \cup D_{l,L_d}$, Invariant 7.10 implies that $|S| \leq 2\delta \cdot |A|$. So the size of the set S is negligible in comparison with the size of the set A. Hence, the size of the set S is negligible even in comparison with the (smaller) set $A \setminus S$. Since $A \setminus S \subseteq B$, it follows that $|S|$ is negligible in comparison with $|B|$ as well. Specifically, we can show that $|S| \leq 4\delta \cdot |B|$. Below, we present the proof in full details.
   Recall that $D_{l,L_d} \subseteq A$ (see Invariant 7.6). Hence, we have:

$$|A| = |D_{l,L_d}| + |A \setminus D_{l,L_d}| \tag{122}$$

Invariant 7.10 states that:

$$|D_{l,L_d}| \leq \delta \cdot |A| \tag{123}$$

Recall that $\delta < 1/2$ (see equation 61). Hence, equations 122 and 123 imply that:

$$|A| \leq \frac{1}{(1-\delta)} \cdot |A \setminus D_{l,L_d}| \leq 2 \cdot |A \setminus D_{l,L_d}| \tag{124}$$

Equations 123 and 124 imply that:

$$|D_{l,L_d}| \leq 2\delta \cdot |A \setminus D_{l,L_d}| \tag{125}$$

Next, Definition 7.2 implies that $|D_c| \leq \delta \cdot |A|$. Accordingly, from equation 124 we get:

$$|D_c| \leq 2\delta \cdot |A \setminus D_{l,L_d}| \tag{126}$$

Equations 125 and 126 imply that:

$$|D_c \cup D_{l,L_d}| \leq 4\delta \cdot |A \setminus D_{l,L_d}| \tag{127}$$

Since $S = D_c \cup D_{l,L_d}$ (see equation 118) and $D_{l0} = D_c \cap A \subseteq D_{l,L_d}$ (see Invariant 7.6), we have:

$$A \setminus S = A \setminus D_{l,L_d} \tag{128}$$

From equations 118, 119, 127 and 128, we conclude that:

$$|S| \leq 4\delta \cdot |B|.$$

5. Equations 118, 119 and Invariant 7.6 imply that:

$$B \setminus S = A \setminus S \subseteq A \setminus D_{l,L_d} = C_{l,L_d}. \tag{129}$$

Corollary 7.9 and equations 121, 129 imply that for all nodes $v \in B \setminus S$, we have:

$$e^{-\gamma} \cdot (\lambda_d L^4/d) \cdot \deg(v,E) \leq \deg(v,X) \leq e^{\gamma} \cdot (\lambda_d L^4/d) \cdot \deg(v,E).$$

6. Since $X = H_{L_d}$ (see equation 121), Corollary 7.24 implies that $\deg(v,X) \leq \lambda_d L^4 + 2$ for all $v \in V$.
7. Equations 118 and 120 imply that:

$$T \setminus S = P \setminus S \subseteq P \setminus D_c \tag{130}$$

By Corollary 7.24 and equations 121, 130, for all nodes $v \in T \setminus S$, we have: $\deg(v,X) \leq 3\varepsilon\lambda_d L^2 + 2$. □

Below, we summarize the main result of this section.

Theorem 7.26. *Suppose that a critical structure (A,P,D_c,H) and a laminar structure $(H_1,\ldots,H_{L_d})$ are maintained as per the algorithm in Section 7.4. Then the edges in the last layer H_{L_d} form the edge-set of a skeleton of $G = (V,E)$.*

Proof. Follows from Definition 5.6 and Lemma 7.25. □

7.6 Bounding the amortized update time of our algorithm

We implement our algorithm using the most obvious data structures. To be more specific, we maintain a doubly linked list for each of the node-sets $A, P, D_c, \{D_{lj}\}$ and $\{C_{lj}\}$. Further, we maintain counters that store the sizes of each of these sets. We also maintain the edge-sets E, H and $\{H_j\}$ using standard adjacency list data structures, and we maintain the degree of each node in each of these edge-sets. Whenever an "element" (an edge or a node) appears in some linked list, we store a pointer from that element to its position in the linked list. Using these pointers, an element can be added to (or deleted from) a list in constant time.

7.6.1 A few notations and terminologies

Recall that initially the graph $G = (V, E)$ has zero edges. We will use the term "edge-update" to denote the insertion/deletion of an edge in $G = (V, E)$. Thus, in the dynamic setting, the graph $G = (V, E)$ changes via a sequence of edge-updates. For any integer $t \geq 0$, we will use the term "edge-update t" to refer to the t^{th} edge-update in this sequence. Each phase corresponds to a contiguous block of edge-updates (see Section 7.4). We can, therefore, identify a phase by an interval $[t, t']$, where $t < t'$ are positive integers. Such a phase begins with the edge-update t and ends just after the edge-update t'. The next phase starts with the edge-update $t' + 1$.

7.6.2 Roadmap

The rest of this section is organized as follows.

- In Section 7.6.3, we show that excluding the calls to the subroutines REVAMP() and REBUILD(j), our algorithm handles each edge-update in $O(L_d)$ time in the worst case.
- In Section 7.6.4, we bound the time taken by a single call to REBUILD(j).
- In Section 7.6.5, we show that the subroutine REVAMP() runs in $O(L^2 L_d/(\varepsilon\delta))$ amortized time (see Lemma 7.33). This includes the time taken by the calls to REBUILD(1) at the end of a phase.
- Finally, in Section 7.6.6, we show that for a given layer $j \in [1, L_d]$, the amortized time taken by a call to REBUILD(j) in the middle of a phase is $O(L^2 L_d^2/(\varepsilon\gamma\delta))$ (see Lemma 7.40). Since there are L_d possible values of j, the total amortized time of all the calls to REBUILD(.) in the middle of a phase is given by $O(L^2 L_d^3/(\varepsilon\gamma\delta))$.

Summing over all these running times, we get our main result.

Theorem 7.27. *Our algorithm for maintaining a critical and a laminar structure handles an edge insertion/deletion in $G = (V, E)$ in $O(L^2 L_d^3/(\varepsilon\gamma\delta))$ amortized time.*

7.6.3 A simple bound

We devote this section to the proof of the following lemma.

Lemma 7.28. *Excluding the calls made to the REVAMP() and REBUILD(j) subroutine, our algorithm handles an edge-update in $G = (V, E)$ in $O(L_d)$ time.*

Proof. Recall the description of our algorithm in Section 7.4. For the purpose of this lemma, we only need to concern ourselves with Section 7.4.2, for Section 7.4.1 only states the initial conditions in the beginning of a phase and Section 7.4.3 describes the REVAMP() subroutine. Accordingly, we analyze the time taken by the Steps I, II and III from Section 7.4.2, one after the other.

- *Running time of Step I.*
  Here, at most one edge is inserted into (or deleted from) the set H, and at most two nodes become c-dirty. It is easy to check that these operations can be implemented in $O(1)$ time.
- *Running time of Step II.*
  This step only asks us to compare the number of c-dirty nodes against the number of active nodes. Hence, this can be implemented in $O(1)$ time.

- *Running time of Step III.*
  Here, we bound the time taken by each of the items (1), (2), (3) and (4) in Step III.
  1. Item (1) moves at most two nodes from D_{lk} to C_{lk} for all $k \in [0, L_d]$, and requires $O(L_d)$ time.
  2. Item (2) inserts/deletes an edge from at most $(L_d + 1)$ edge-sets $\{H_j\}$, and requires $O(L_d)$ time.
  3. Item (3) makes two calls to CLEANUP(x). Each of these calls runs in $O(L_d)$ time.
  4. Item (4) calls VERIFY(), which requires $O(L_d)$ time excluding the call to REBUILD(j).

  Hence, the total time taken by Step III is $O(L_d)$.

Hence, excluding the calls to REBUILD(j) and REVAMP(), handling an edge-update requires $O(L_d)$ time. □

7.6.4 Analyzing the running time of a single call to REBUILD(j)

We will bound the running time of a call to the subroutine REBUILD(j) as described in Figure 3. But first, recall the initial conditions stated in Figure 2 and the properties of the subroutine stated in Lemma 7.12.

Lemma 7.29. *Consider any layer $j \in [1, L_d]$, and note that the subroutine REBUILD(j) does not change the set of active nodes. A call to the subroutine REBUILD(j) runs in $O(|A| \cdot d \cdot 2^{-j})$ time.*

Proof. The call to REBUILD(j) does not affect the layers $j' < j$. Instead, the subroutine iteratively sets $H_{j'} \leftarrow \text{SPLIT}(H_{j'-1})$ for $j' = j$ to L_d. See Figure 3.

Consider the first iteration of the For loop in Figure 3, where we set $H_j \leftarrow \text{SPLIT}(H_{j-1})$. Since $H_{j-1} \subseteq H$ (see Definition 7.4), and H is the set of edges incident upon active nodes (see Definition 7.2), each edge in H_{j-1} has at least one endpoint in the set A. Also by Lemma 7.23, the maximum degree of a node in a layer j' is $O(d \cdot 2^{-j'})$. Hence, there are at most $O(|A| \cdot d \cdot 2^{-(j-1)})$ edges in H_{j-1}. Thus, we have:

$$|H_{j-1}| = O(|A| \cdot d \cdot 2^{-(j-1)}). \tag{131}$$

By Figure 1, we can implement the subroutine SPLIT(H_{j-1}) in $O(|H_{j-1}|)$ time. After setting $H_j \leftarrow \text{SPLIT}(H_{j-1})$, we have to perform the following operations: (a) $D_{l,j} \leftarrow D_{l,j-1}$ and (b) $C_{l,j} \leftarrow C_{l,j-1}$. Since both the sets $D_{l,j-1}$ and $C_{l,j-1}$ are contained in A (see Invariant 7.6), these operations can be performed in $O(|A|)$ time. Since $j \leq L_d$, we have $d \cdot 2^{-(j-1)} \geq 1$ (see equation 58). Thus, the time taken by the first iteration of the For loop in Figure 3 is $O(|H_{j-1}| + |A|) = O(|A| \cdot d \cdot 2^{-(j-1)})$.

Applying the same argument by which we arrived at equation 131, immediately after setting $H_j \leftarrow \text{SPLIT}(H_{j-1})$, we have the following guarantee:

$$|H_j| = O(|A| \cdot d \cdot 2^{-j}) \tag{132}$$

Accordingly, the second iteration of the For loop in Figure 3, where we set $H_{j+1} \leftarrow \text{SPLIT}(H_j)$, can be implemented in $O(|A| \cdot d \cdot 2^{-j})$ time. In general, it can be shown that setting $H_k \leftarrow \text{SPLIT}(H_{k-1})$ will take $O(|A| \cdot d \cdot 2^{-(k-1)})$ time for $k \in [j, L_d]$. So the runtime of the subroutine REBUILD(j) is given by:

$$\sum_{k=j}^{L_d} O\left(|A| \cdot d \cdot 2^{-(k-1)}\right) = O\left(|A| \cdot d \cdot 2^{-j}\right).$$

This concludes the proof of Lemma 7.29. □

7.6.5 Bounding the amortized update time of the subroutine REVAMP()

Recall the notations and terminologies introduced in Section 7.6.1. Our algorithm works in "phases" (see the discussion in the beginning of Section 7.4). For every integer $k \geq 1$, suppose that the phase k starts with the edge-update t_k. Thus, we have $t_1 = 1$ and $t_k < t_{k+1}$ for every integer $k \geq 1$, and each phase k corresponds to the interval $[t_k, t_{k+1} - 1]$. Next, note that a call is made to the subroutine REVAMP() at the end of each phase (sec Section 7.4.3). So the k^{th} call to the subroutine REVAMP() occurs while processing the edge-update $t_{k+1} - 1$.

- Throughout the rest of this section, we will use the properties of our algorithm outlined in Lemma 7.21. We introduce the following notations.
  - In the beginning of a phase $k \geq 1$, we have $D_c = \emptyset$. As the phase progresses, the set D_c keeps getting bigger and bigger. Finally, due to the edge-update $t_{k+1} - 1$, the size of the set D_c exceeds the threshold $(\delta/(L_d+1)) \cdot |A|$. Let $D_c^+(k)$ denote the status of the set D_c precisely at this instant. Note that immediately afterwords, a call is made to the subroutine REVAMP(), and when the subroutine finishes execution we again find that the set D_c is empty. Then we initiate the next phase.
  - The indicator variable $\textsc{Dirty}_c^+(v,k) \in \{0,1\}$ is set to one iff $v \in D_c^+(k)$. Thus, we have:

$$\left|D_c^+(k)\right| = \sum_{v \in V} \textsc{Dirty}_c^+(v,k) \tag{133}$$

  - The sets A and P do not change in the middle of a phase. Hence, without any ambiguity, we let $A(k)$ and $P(k)$ respectively denote the status of the set A (resp. P) during phase $k \geq 1$.
  - Consider a counter $\mathscr{U}_v$. Initially, the graph $G = (V,E)$ is empty and $\mathscr{U}_v = 0$. Subsequently, whenever an edge-update occurs in the graph $G = (V,E)$, if the edge being inserted/deleted is incident upon the node v, then we set $\mathscr{U}_v \leftarrow \mathscr{U}_v + 1$.
  - Let $\mathscr{U}_v(t)$ denote the status of $\mathscr{U}_v$ after the edge-update $t \geq 0$. Note that $\mathscr{U}_v(0) = 0$, and in general, $\mathscr{U}_v(t)$ gives the number of edge-updates that are incident upon v among the first t edge-updates.

Roadmap. In Lemma 7.30 and Corollary 7.31, we bound the time taken by the call to REVAMP() at the end of a given phase $k \geq 1$. Specifically, we show that such a call runs in $O(|D_c^+(k)| \cdot d \cdot L_d \cdot \delta^{-1})$ time. In Lemma 7.32, we show that on average, a node v can become c-dirty only after $\Omega(\varepsilon d L^{-2})$ edge-updates incident upon v have taken place in the graph $G = (V,E)$. By Corollary 7.31, processing a c-dirty node at the end of the phase requires $O(dL_d\delta^{-1})$ time. Hence, we get an amortized update time of $O((dL_d\delta^{-1})/(\varepsilon dL^{-2})) = O(L^2 L_d/(\varepsilon\delta))$ for the subroutine REVAMP(). This is proved in Lemma 7.33.

Lemma 7.30. *The call to REVAMP() at the end of phase k runs in $O(|D_c^+(k) \cup A(k)| \cdot d)$ time.*

Proof. Recall the description of the subroutine REVAMP() in Section 7.4.3. The node-set V' in the For loop is given by the set $D_c^+(k)$. We first bound the running time of this For loop (see item (1) in Section 7.4.3).

During a single iteration of the For loop, we pick one c-dirty node v and make it c-clean. Next, we might have to change the status of the node from active to passive (or vice versa) and remove (resp. add) the edges incident upon v from (resp. to) the set H. Since the node v has at most d edges incident upon it (see Lemma 7.1), one iteration of the For loop can be implemented in $O(d)$ time. Since the For loop runs for $D_c^+(k)$ iterations, the total time taken by the For loop is $O(|D_c^+(k)| \cdot d)$.

Next, we bound the time taken to implement item (2) in Section 7.4.3. In the beginning of the For loop, we had $|A(k)|$ many active nodes. In the worst case, the For loop can potentially change the status of every passive node in $D_c^+(k)$ to active. Thus, there are at most $|D_c^+(k) \cup A(k)|$ many active nodes at the end of the For loop. So the first part of item (2), where we update the edge-sets $\{H_j\}$ and the node-sets $\{D_{lj}, C_{lj}\}$, can be implemented in $O(|D_c^+(k) \cup A(k)| \cdot d)$ time. Next, by Lemma 7.29, the call to REBUILD(1) also takes $O(|D_c^+(k) \cup A_k| \cdot d)$ time.

Thus, the total time taken by call to REVAMP() at the end of phase k is $O(|D_c^+(k) \cup A(k)| \cdot d)$. □

Corollary 7.31. *The call to REVAMP() at the end of phase k runs in $O(|D_c^+(k)| \cdot d \cdot (L_d/\delta))$ time.*

Proof. Follows from Lemma 7.30 and the fact that $|A(k)| = O((L_d/\delta) \cdot |D_c^+(k)|)$ (see Lemma 7.21). □

We now explain the statement of Lemma 7.32. First, note that a node can change its status from c-clean to c-dirty at most once during the course of a single phase (see Lemma 7.21). Accordingly, consider any node $v \in V$, and let $\psi(k) = \sum_{k'=1}^{k} \textsc{Dirty}_c^+(v,k')$ be the number of times the node v becomes c-dirty

during the first k phases, where $k \geq 1$ is an integer. Then at least $(2\varepsilon d/L^2) \cdot \psi(k)$ edge-updates in the first k phases were incident upon v. To summarize, this lemma shows that on average one needs to have $(2\varepsilon d/L^2)$ edge-updates incident upon v before the node v changes its status from c-clean to c-dirty.

Lemma 7.32. *Fix any node $v \in V$ and any positive integer k. We have:*

$$\mathscr{U}_v(t_{k+1}-1) \geq \left(\frac{2\varepsilon d}{L^2}\right) \cdot \sum_{k'=1}^{k} \textsc{Dirty}_c^+(v,k') \tag{134}$$

Proof. We will use induction on the value of the sum $\psi(k) = \sum_{k'=1}^{k} \textsc{Dirty}_c^+(v,k')$. Specifically, throughout the proof we fix a phase $k \geq 1$. Next, we define $\psi(k) = \sum_{k'=1}^{k} \textsc{Dirty}_c^+(v,k')$, and consider the last edge-update t in the interval $[t_1, t_{k+1}-1]$ where the node v becomes c-dirty from c-clean. We will show that $\mathscr{U}_v(t) \geq (2\varepsilon d/L^2) \cdot \psi(k)$. Since $t_{k+1}-1 \geq t$, this will imply that $\mathscr{U}_v(t_{k+1}-1) \geq \mathscr{U}_v(t) \geq (2\varepsilon d/L^2) \cdot \psi(k)$.

The base step. $\psi(k) = 1$.

In the beginning of phase one, the graph $G = (V,E)$ is empty, $\deg(v,E) = 0$, and the node v is passive and c-clean. Since $\psi(k) = 1$, there is exactly one phase $k' \in [1,k]$ such that $v \in D_c^+(k)$. Recall the description of our algorithm in Section 7.4.2. It follows that $\deg(v,E)$ was less than the threshold $(3\varepsilon d/L^2)$ from the start of phase one till the end of phase $(k'-1)$. It was only after the edge-update t in the middle of phase k' that we saw $\deg(v,E)$ reaching the threshold $(3\varepsilon d/L^2)$, which compelled us to classify the node v as c-dirty at that instant. Since $\deg(v,E)$ was initially zero, at least $(3\varepsilon d/L^2)$ edges incident upon v must have been inserted into $G = (V,E)$ during the first t edge-updates. Hence, we must have $\mathscr{U}_v(t) \geq (3\varepsilon d/L^2) = (3\varepsilon d/L^2) \cdot \psi(k) > (2\varepsilon d/L^2) \cdot \psi(k)$.

The inductive step. $\psi(k) \geq 2$.

Consider the phase $k^* < k$ where the node v becomes c-dirty for the $(\psi(k)-1)^{th}$ time. Specifically, let $k^* = \min\{k' \in [1,k-1] : \sum_{j=1}^{k'} \textsc{Dirty}_c^+(v,j) = \psi(k)-1\}$. Let t^* be the edge-update in phase k^* due to which the node v becomes c-dirty from c-clean. By induction hypothesis, we have:

$$\mathscr{U}_v(t^*) \geq \left(\frac{2\varepsilon d}{L^2}\right) \cdot (\psi(k)-1) \tag{135}$$

Consider the interval $[t_{k^*+1}, t_{k+1}-1]$ from the start of phase (k^*+1) till the end of phase k. Let t be the unique edge-update in this interval where the node v becomes c-dirty from c-clean. We will show that at least $(2\varepsilon d/L^2)$ edge-updates incident upon v took place during the interval $[t^*+1, t]$. Specifically, we will prove the following inequality.

$$\mathscr{U}_v(t) - \mathscr{U}_v(t^*) \geq \left(\frac{2\varepsilon d}{L^2}\right) \tag{136}$$

Equations 135 and 136 will imply that $\mathscr{U}_v(t) \geq (2\varepsilon d/L^2) \cdot \psi(k)$. To proceed with the proof of equation 136, note that there are two possible ways by which the node v could have become c-dirty at edge-update t^*, and accordingly, we consider two possible cases.

- *Case 1. In the beginning of phase k^*, the node v was passive and c-clean, with $\deg(v,E) < (3\varepsilon d/L^2)$. Then with the edge-update t^* in the middle of phase k^*, we saw $\deg(v,E)$ reaching the threshold $(3\varepsilon d/L^2)$, which forced us to classify the node v as c-dirty at that instant.*
  In this instance, we need to consider two possible sub-cases.
  - *Case 1a. At the end of phase k^*, i.e., immediately after edge-update $(t_{k^*+1}-1)$, the degree of the node v is at most $(\varepsilon d/L^2)$.*

In this case, during the interval $[t^*, t_{k^*+1}-1]$ itself the degree of the node v has changed from $(3\varepsilon d/L^2)$ to $(\varepsilon d/L^2)$. This can happen only if $(3\varepsilon d/L^2)-(\varepsilon d/L^2)=(2\varepsilon d/L^2)$ edges incident upon v was deleted from the graph during this interval. Since $t_{k^*+1}-1<t$, we get:

$$\mathscr{U}_v(t)-\mathscr{U}_v(t^*)\geq \mathscr{U}_v(t_{k^*+1}-1)-\mathscr{U}_v(t^*)\geq \left(\frac{2\varepsilon d}{L^2}\right).$$

– *Case 1b. At the end of phase k^*, i.e., immediately after edge-update $(t_{k^*+1}-1)$, the degree of the node v is greater than $(\varepsilon d/L^2)$.*
In this case, the node v is classified as active in phase k^*+1 (see Section 7.4.3). Further, the node v is c-clean at the start of phase k^*+1 (see Lemma 7.21) and remains c-clean throughout the interval $[t_{k^*+1}, t-1]$ (this follows from our choice of t). Accordingly, the node remains active throughout the interval $[t_{k^*+1}, t-1]$ (see Section 7.4.3). To summarize, the node remains active and c-clean from the start of phase k^*+1 till the edge-update $t-1$.
By definition, after the edge-update t the node v becomes c-dirty again. An active node becomes c-dirty only if its degree reaches the threshold $(\varepsilon d/L^2)$. Thus, we must have $\deg(v,E)=\varepsilon d/L^2$ after edge-update t. Since $\deg(v,E)=(3\varepsilon d/L^2)$ after edge-update t^*, we infer that $\deg(v,E)$ has dropped by $(3\varepsilon d/L^2)-(\varepsilon d/L^2)=(2\varepsilon d/L^2)$ during the interval $[t^*,t]$. This can happen only if at least $(2\varepsilon d/L^2)$ edges incident upon v were deleted from the graph $G=(V,E)$ during the interval $[t^*,t]$. Thus, we get:

$$\mathscr{U}_v(t)-\mathscr{U}_v(t^*)\geq (2\varepsilon d/L^2).$$

- *Case 2. In the beginning of phase k^*, the node v was active and c-clean, with $deg(v,E)>(\varepsilon d/L^2)$. Then with the edge-update t^* in the middle of phase k^*, we saw $deg(v,E)$ reaching the threshold $(\varepsilon d/L^2)$, which forced us to classify the node v as c-dirty at that instant.*
We can easily modify the proof of Case 1 for this case. Basically, the roles of the active and passive nodes are interchanged, and so are the roles of the thresholds $(3\varepsilon d/L^2)$ and $(\varepsilon d/L^2)$.

□

We are now ready to bound the amortized update time of the subroutine REVAMP().

Lemma 7.33. *The subroutine REVAMP() has an amortized update time of $O(L^2 L_d/(\varepsilon\delta))$.*

Proof. Recall that $\mathscr{U}_v(t)$ denotes the number of edge-updates incident upon v among the first t edge-updates in $G=(V,E)$. Since each edge is incident upon two nodes, we have $t=(1/2)\cdot\sum_{v\in V}\mathscr{U}_v(t)$ for every integer $t\geq 1$. Hence, we have the following guarantee at the end of every phase $k\geq 1$.

$$\begin{aligned}
t_{k+1}-1 &= (1/2)\cdot\sum_{v\in V}\mathscr{U}_v(t_{k+1}-1)\\
&\geq (\varepsilon d/L^2)\cdot\sum_{v\in V}\sum_{k'=1}^{k}\textsc{Dirty}_c^+(v,k') \qquad (137)\\
&= (\varepsilon d/L^2)\cdot\sum_{k'=1}^{k}\sum_{v\in V}\textsc{Dirty}_c^+(v,k')\\
&= (\varepsilon d/L^2)\cdot\sum_{k'=1}^{k}\left|D_c^+(v,k')\right| \qquad (138)
\end{aligned}$$

Equation 137 follows from Lemma 7.32. Equation 138 follows from equation 133. From equation 138 and Corollary 7.31, we get the following guarantee.

- The total time spent on the calls to the subroutine REVAMP() during the first k phases is given by:

$$\sum_{k'=1}^{k} O(|D_c^+(k)| \cdot d \cdot (L_d/\delta)) = (t_{k+1} - 1) \cdot O(L^2 L_d/(\varepsilon\delta)).$$

Note that by definition, there are $(t_{k+1} - 1)$ edge-updates in the graph $G = (V, E)$ during the first k phases. Accordingly, we infer that the subroutine REVAMP() has an amortized update time of $O(L^2 L_d/(\varepsilon\delta))$. □

7.6.6 Bounding the amortized update time of REBUILD(j) in the middle of a phase

Throughout this section, we fix any layer $j \in [1, L_d]$. We will analyze the amortized running time of a call made to the subroutine REBUILD(j). Note that such a call can be made under two possible circumstances.

- While processing an edge-update in the middle of a phase, a call to REBUILD(j) can be made by the subroutine VERIFY() in Figure 5. This happens only if there are too many l-dirty nodes at layer j.
- While processing the last edge-update of a phase, the subroutine REVAMP() calls REBUILD(j) with $j = 1$. The total running time of these calls to REBUILD(1) is subsumed by the total running time of all the calls to REVAMP(), which in turn has already been analyzed in Section 7.6.5.

Accordingly, in this section we focus on the calls made to REBUILD(j) from a given phase.

- Throughout the rest of this section, we will use the properties outlined in Lemmas 7.21 and 7.22. Further, since we are considering a fixed layer $j \in [1, L_d]$, we will sometimes omit the symbol "j" from the notations introduced in this section.

Before proceeding any further, we need to introduce the concept of an "epoch".

Definition 7.34. *An epoch $[\tau^0, \tau^1]$ consists of a contiguous block of edge-updates in the middle of a phase, where τ^0 (resp. τ^1) denotes the first (resp. last) edge-update in the epoch. Specifically, we have:*

- *While processing the edge-update $(\tau^0 - 1)$, a call is made to REBUILD(j') with $j' \in [1, j]$.*
- *While processing the edge-updates $t \in [\tau^0, \tau^1 - 1]$, no call is made to REBUILD(j') with $j' \in [1, j]$.*
- *While processing the edge-update τ^1, a call is made to REBUILD(j).*

We say that the epoch "begins" at the time instant just before the edge-update τ^0, and "ends" at the time instant just before the call to REBUILD(j) while processing the edge-update τ^1.

For the rest of this section, we fix a given epoch $[\tau^0, \tau^1]$ in the middle of the phase under consideration. By Lemma 7.29, the call to REBUILD(j) after edge-update τ^1 takes $O(|A| \cdot d \cdot 2^{-j})$ time. We will show that the epoch lasts for $\Omega(\varepsilon\gamma\delta \cdot (L_d L)^{-2} \cdot |A| \cdot (d/2^j))$ edge-updates (see Corollary 7.39). Note that by definition, no call is made to REBUILD(j) during the epoch. Hence, dividing the running time of the call that ends the epoch by the number of edge-updates in the epoch gives us an amortized bound of $O(L_d^2 L^2/\varepsilon\gamma\delta)$.

Thus, the main challenge is to lower bound the number of edge-updates in the epoch. Towards this end, we take a closer look at how the sets of l-dirty nodes at layers $j' \in [0, j]$ evolve with the passage of time. First, note the properties of the laminar structure specified by Invariant 7.6. Since a call was made to REBUILD(j') with $j' \in [1, j]$ after the edge-update $(\tau^0 - 1)$, Lemma 7.12 implies that $D_{l0} \subseteq D_{l1} \subseteq \cdots \subseteq D_{l,j-1} = D_{l,j} \subseteq A$ when the epoch begins. In the middle of the epoch, the sets $\{D_{l,k}\}, k \in [0, j]$, can only get bigger with the passage of time (see Lemma 7.22), but they always remain contained within one another as a laminar family. To be more specific, suppose that while processing some edge-update in the epoch, we have to change the sets $D_{l0}, \ldots, D_{lj}$. This change has to be one of the following two types.

- (a) Some node $v \in A$ becomes c-dirty, and to satisfy Invariant 7.6 we move the node v from C_{lk} to D_{lk} at all layers $k \in [0, j]$. See item (1) in Step III of Section 7.4.2.
- (b) Some node $v \in C_{lj'}$ becomes l-dirty at layer $j' \in [1, j]$, and we move the node v from C_{lk} to D_{lk} at all layers $k \in [j', j]$. See the call to CLEANUP(x) in Step III of Section 7.4.2.

To summarize, during the epoch the l-dirty sets $D_{l0} \subseteq D_{l1} \subseteq \cdots \subseteq D_{lj}$ keep getting bigger and bigger with the passage of time, without violating the laminar property. In other words, no node is ever deleted from one of these sets while the epoch is still in progress.

The epoch ends because a call is made to REBUILD(j) after edge-update τ^1. This can happen only if at the end of the epoch, Invariant 7.10 is violated at layer j but satisfied at all the layers $k \in [0, j-1]$. See the call to the subroutine VERIFY() in Step III (Figure 5) of Section 7.4.2. Thus, at the end of the epoch, we have $|D_{lj}| > (\delta(j+1)/(L_d+1)) \cdot |A|$ and $|D_{l,j-1}| \leq (\delta j/(L_d+1)) \cdot |A|$. Since $D_{l,j-1} \subseteq D_{l,j}$, we get:

Lemma 7.35. *At the end of the epoch $[\tau^0, \tau^1]$, we have $|D_{lj} \setminus D_{l,j-1}| \geq (\delta/(L_d+1)) \cdot |A|$.*

Next, note that if a node v belongs to $D_{lj} \setminus D_{l,j-1}$ at the end of the epoch, then the node v must have been part of the set C_{lj} in the beginning of the epoch. This follows from the three observations stated below, which, in turn, follow from our discussion preceding Lemma 7.35.

1. We always have $D_{l,j-1} \subseteq D_{lj} \subseteq A$ and $C_{lj} = A \setminus D_{lj}$. The set A does not change during the epoch.
2. In the beginning of the epoch, we have $D_{l,j-1} = D_{lj}$.
3. During the epoch, a node is never deleted from either of the sets $D_{l,j-1}$ and $D_{l,j}$.

We formally state our observation in Lemma 7.36.

Lemma 7.36. *If a node v belongs to $D_{lj} \setminus D_{l,j-1}$ at the end of the epoch $[\tau^0, \tau^1]$, then the node v must have belonged to C_{lj} in the beginning of the epoch.*

The next lemma follows directly from Lemmas 7.35 and 7.36.

Lemma 7.37. *There are at least $\Omega(\delta \cdot L_d^{-1} \cdot |A|)$ nodes that belong to C_{lj} when the epoch $[\tau^0, \tau^1]$ begins and belong to $D_{lj} \setminus D_{l,j-1}$ when the epoch ends.*

Next, we will show that a node v moves from C_{lj} to $D_{lj} \setminus D_{l,j-1}$ only after a large number of edge-updates incident upon v. The complete proof of Lemma 7.38 appears in Section 7.6.7. The main idea behind the proof, however, is simple. Consider a node v that belongs to C_{lj} when the epoch begins. Since a call was made to REBUILD(j') with $j' \in [1, j]$ just before the start of the epoch, Lemma 7.11 implies that $\deg(v, H_j)$ is very close to $(1/2) \cdot \deg(v, H_{j-1})$ at that moment. On the other hand, the node v moves from C_{lj} to $D_{lj} \setminus D_{l,j-1}$ only if somewhere in the middle of the epoch it violates Invariant 7.7 in layer j, and this means that at that moment $\deg(v, H_j)$ is very far away from $(1/2) \cdot \deg(v, H_{j-1})$. So $\deg(v, H_j)$ moves from being close to $(1/2) \cdot \deg(v, H_{j-1})$ to being far away from $(1/2) \cdot \deg(v, H_{j-1})$ within the given epoch. This can happen only if either $\deg(v, H_j)$ or $\deg(v, H_{j-1})$ changes by a large amount during the epoch. In either case, we can show that a large number of edge-updates incident upon v takes place during the epoch.

Lemma 7.38. *Take any node v that is part of C_{lj} when the epoch $[\tau^0, \tau^1]$ begins, and is part of $D_{lj} \setminus D_{l,j-1}$ when the epoch ends. At least $\Omega(\varepsilon\gamma \cdot (L_d L^2)^{-1} \cdot (d/2^j))$ edge-updates in the epoch are incident upon v.*

Corollary 7.39. *The epoch $[\tau^0, \tau^1]$ lasts for at least $\Omega(\varepsilon\gamma\delta \cdot (L_d L)^{-2} \cdot (d/2^j) \cdot |A|)$ edge-updates.*

Proof. Let C^* be the set of nodes that are part of C_{lj} when the epoch begins and part of $D_{lj} \setminus D_{l,j-1}$ when the epoch ends. By Lemmas 7.37, 7.38, at least $\Omega(\varepsilon\gamma\delta \cdot L_d^{-2} \cdot L^{-2} \cdot |A| \cdot d \cdot 2^{-j})$ edge-updates in the epoch are incident upon the nodes in C^*. This lower bounds the total number of edge-updates during the epoch. □

We are now ready to derive the main result of this section.

Lemma 7.40. *The amortized update time of REBUILD(j) in the middle of a phase is $O(L^2 L_d^2/\varepsilon\gamma\delta)$.*

Proof. Note that no call is made to REBUILD(j) during the epoch, and the call to REBUILD(j) after edge-update τ^1 requires $O(|A| \cdot d \cdot 2^{-j})$ time (see Lemma 7.29). The lemma now follows from Corollary 7.39. □

7.6.7 Proof of Lemma 7.38

Notations. We introduce some notations that will be used throughout the proof.

- Let x_k be the value of $\deg(v, H_k)$, for $k \in [0, L_d]$, when the epoch begins. Similarly, let x be the value of $\deg(v, E)$ when the epoch begins.
- Consider the unique time-instant in the epoch when the node v is moved from $C_{l,j}$ to $D_{lj} \setminus D_{l,j-1}$. For $k \in [0, L_d]$, let $x_k + \Delta_k$ be the value of $\deg(v, H_k)$ at this time-instant, where Δ_k is some integer.

Two simple observations. Just before the epoch begins, a call is made to REBUILD(j') for some $j' \in [1, j]$. At the end of the call, $\deg(v, H_j)$ equals $1/2$ times $\deg(v, H_{j-1})$, plus-minus one (see Lemma 7.11). Thus, we have:

$$\left(\frac{x_{j-1}}{2}\right) - 1 \leq x_j \leq \left(\frac{x_{j-1}}{2}\right) + 1 \tag{139}$$

Consider the unique time-instant in the epoch when the node v is moved from C_{lj} to $D_{lj} \setminus D_{l,j-1}$. This event can take place only if the node v was violating Invariant 7.7 at layer j at that instant. Thus, we have:

$$x_j + \Delta_j \notin \left[(1+\gamma/L_d)^{-1}\left(\frac{x_{j-1}+\Delta_{j-1}}{2}\right), (1+\gamma/L_d)\left(\frac{x_{j-1}+\Delta_{j-1}}{2}\right)\right] \tag{140}$$

The main idea. When the epoch begins, equation 139 implies that $\deg(v, H_j)$ is very close to $(1/2) \cdot \deg(v, H_{j-1})$. In contrast, equation 140 implies that somewhere in the middle of the epoch, $\deg(v, H_j)$ is quite far away from $(1/2) \cdot \deg(v, H_{j-1})$. Intuitively, this can happen only if either $\deg(v, H_j)$ or $\deg(v, H_{j-1})$ has changed by a large amount during this interval, i.e., either $|\Delta_j|$ or $|\Delta_{j-1}|$ is large. This is shown in Claim 7.43 and Corollary 7.44. Finally, in Claim 7.45 and Corollary 7.46, we show that this can happen only if the degree of v in the graph $G = (V, E)$ itself has changed by a large amount, which means that a large number of edge-updates incident upon v have taken place during the interval under consideration. Lemma 7.38 follows from Corollary 7.46.

To prove Claim 7.43 and Corollary 7.44, we first need to show that x_{j-1} is not much smaller than $d/2^j$.

Claim 7.41. *We have* $x_{j-1} \geq (2\varepsilon \cdot e^{-\gamma} L^{-2}) \cdot (d/2^j)$.

Proof. When the epoch begins, the node v belongs to the set C_{lj}, and we know that $C_{lj} \subseteq C_{l,j-1}$ (see Invariant 7.6). Hence, when the epoch begins, the node v is part of $C_{l,j-1}$. Accordingly, by Lemma 7.8:

$$x_{j-1} \geq \frac{x}{2^{j-1} \cdot (1+\gamma/L_d)^{j-1}} \tag{141}$$

We now lower bound x. When the epoch begins, the node v belongs to the set C_{lj}, and we know that $C_{lj} \subseteq A \setminus D_{l0} = A \setminus D_c$ (see Invariant 7.6). Hence, by the condition (1) in Definition 7.2:

$$x > \varepsilon d / L^2 \tag{142}$$

From equations 141 and 142 we infer that:

$$x_{j-1} \geq \frac{\varepsilon \cdot d}{2^{j-1} \cdot (1+\gamma/L_d)^{j-1} \cdot L^2} \tag{143}$$

Since $(1+\gamma/L_d)^{j-1} < (1+\gamma/L_d)^{L_d} \leq e^{\gamma}$, equation 143 implies that:

$$x_{j-1} \geq \left(\frac{\varepsilon}{e^{\gamma} L^2}\right) \cdot \left(\frac{d}{2^{j-1}}\right) = \left(\frac{2\varepsilon}{e^{\gamma} L^2}\right) \cdot \left(\frac{d}{2^j}\right)$$

This concludes the proof of the claim. □

Corollary 7.42. *We have* $x_{j-1} \geq 8L_d/\gamma$.

Proof. Form Claim 7.41, we infer that:

$$\begin{aligned} x_{j-1} &\geq \left(\frac{\varepsilon}{e^{\gamma}L^2}\right)\cdot\left(\frac{d}{2^{j-1}}\right) \\ &\geq \left(\frac{\varepsilon}{e^{\gamma}L^2}\right)\cdot\left(\frac{d}{2^{L_d}}\right) && (144) \\ &= \frac{\varepsilon\cdot\lambda_d\cdot L^2}{e^{\gamma}} && (145) \\ &\geq \frac{8L_d}{\gamma} && (146) \end{aligned}$$

Equation 144 holds since $j \leq L_d$. Equation 145 follows from equation 59. Equation 146 follows from equation 68. □

Claim 7.43. *Either* $|\Delta_j| \geq \gamma\cdot(16L_d)^{-1}\cdot x_{j-1}$ *or* $|\Delta_{j-1}| \geq \gamma\cdot(16L_d)^{-1}\cdot x_{j-1}$.

Proof. Throughout the proof, we set $\mu = (1+\gamma/L_d)$. Looking at equation 140, we consider two cases.

Case 1. $x_j+\Delta_j > (1/2)\cdot\mu\cdot(x_{j-1}+\Delta_{j-1})$.

In this case, since $x_j \leq x_{j-1}/2+1$ (see equation 139), we get:

$$\Delta_j \geq \mu\cdot\left(\frac{x_{j-1}+\Delta_{j-1}}{2}\right)-\left(\frac{x_{j-1}}{2}\right)-1 \geq \frac{\mu\cdot\Delta_{j-1}}{2}+(\mu-1)\cdot\left(\frac{x_{j-1}}{2}\right)-1$$

Rearranging the terms in the above inequality, we get:

$$\begin{aligned} \Delta_j-\frac{\mu\cdot\Delta_{j-1}}{2} &\geq (\mu-1)\cdot\left(\frac{x_{j-1}}{2}\right)-1 \\ &\geq (\mu-1)\cdot\left(\frac{x_{j-1}}{2}\right)-(\mu-1)\cdot\left(\frac{x_{j-1}}{4}\right) && (147) \\ &= (\mu-1)\cdot\left(\frac{x_{j-1}}{4}\right) && (148) \end{aligned}$$

Equation 147 holds since $x_{j-1} > 8L_d/\gamma$ (see Corollary 7.42) and $(\mu-1) = \gamma/L_d$. Thus, from equation 148, we infer that:

$$|\Delta_j|+\left|\frac{\mu\cdot\Delta_{j-1}}{2}\right| \geq (\gamma/L_d)\cdot\left(\frac{x_{j-1}}{4}\right) \quad (149)$$

Hence, either $|\Delta_j|$ is at least $1/2$ times the right hand side of equation 149, or else $|\Delta_{j-1}|$ is at least $1/\mu$ times the right hand side of equation 149. The claim follows since $\mu = 1+\gamma/L_d \leq 2$.

Case 2. $x_j+\Delta_j < (1/2)\cdot(1/\mu)\cdot(x_{j-1}+\Delta_{j-1})$.

In this case, since $x_j \geq x_{j-1}/2-1$ (see equation 139), we get:

$$\Delta_j \leq \left(\frac{x_{j-1}+\Delta_{j-1}}{2\mu}\right)-\left(\frac{x_{j-1}}{2}\right)+1 \leq \left(\frac{\Delta_{j-1}}{2\mu}\right)+(1/\mu-1)\cdot\left(\frac{x_{j-1}}{2}\right)+1$$

Rearranging the terms in the above inequality, we get:

$$\begin{aligned} \left(\frac{\Delta_{j-1}}{2\mu}\right)-\Delta_j &\geq (1-1/\mu)\cdot\left(\frac{x_{j-1}}{2}\right)-1 \\ &\geq \left(\frac{\mu-1}{\mu}\right)\cdot\left(\frac{x_{j-1}}{2}\right)-(\mu-1)\cdot\left(\frac{x_{j-1}}{4}\right) && (150) \\ &\geq \left(\frac{\mu-1}{\mu}\right)\cdot\left(\frac{x_{j-1}}{4}\right) && (151) \end{aligned}$$

Equations 150 and 151 hold since $\mu = 1+\gamma/L_d$ and $x_{j-1} > 8L_d/\gamma = 8/(\mu-1)$ (see Corollary 7.42). From equation 151, we infer that:

$$|\Delta_j| + \left|\frac{\Delta_{j-1}}{2\mu}\right| \geq \left(\frac{\mu-1}{\mu}\right)\cdot\left(\frac{x_{j-1}}{4}\right) = (\gamma/L_d)\cdot\left(\frac{x_{j-1}}{4\mu}\right) \tag{152}$$

Since $\mu = 1+\gamma/L_d \leq 2$, from equation 152 we have:

$$|\Delta_j| + \left|\frac{\Delta_{j-1}}{2\mu}\right| \geq (\gamma/L_d)\cdot\left(\frac{x_{j-1}}{8}\right) \tag{153}$$

Hence, either $|\Delta_j|$ is at least $1/2$ times the right hand side of equation 149, or else $|\Delta_{j-1}|$ is at least μ times the right hand side of equation 149. The claim follows since $\mu = 1+\gamma/L_d \geq 1$. □

Corollary 7.44. *Either* $|\Delta_j| \geq \varepsilon\gamma\cdot(8e^{\gamma}L_dL^2)^{-1}\cdot(d/2^j)$ *or* $|\Delta_{j-1}| \geq \varepsilon\gamma\cdot(8e^{\gamma}L_dL^2)^{-1}\cdot(d/2^j)$.

Proof. Follows from Claims 7.41 and 7.43. □

Claim 7.45. *At least* $\max(|\Delta_j|,|\Delta_{j-1}|)$ *edge-updates in the epoch* $\left[\tau^1,\tau^0\right]$ *are incident upon the node v.*

Proof. The proof consists of two steps.

- *Step 1. We show that at least* $|\Delta_j|$ *edge-updates in the epoch are incident upon the node v.*
  To prove this step, note that $j \in [1,L_d]$. By definition, no call is made to REBUILD(j') with $j' \in [1,j]$ during the epoch. Hence, no edge is inserted into H_j during the epoch. Further, during the epoch, an edge $e \in H_j$ can get deleted from H_j only if the edge gets deleted from the graph $G=(V,E)$ itself (see Lemma 7.22). Thus, during any interval within the epoch $\deg(v,H_j)$ can only decrease, and furthermore, the absolute value of this change in $\deg(v,H_j)$ is at most the number edge-updates incident upon v.
- *Step 2. We show that at least* $|\Delta_{j-1}|$ *edge-updates in the epoch are incident upon the node v.*
  Here, we consider two possible cases.
  - *Case 1.* $j-1 \in [1,L_d]$.
    In this case, the argument is exactly similar to Case 1.
  - *Case 2.* $j-1=0$.
    In this case, since the node v is active, we have $\deg(v,H_0) = \deg(v,H) = \deg(v,E)$ throughout the duration of the epoch. Hence, during any interval within the epoch, the absolute value of the change in $\deg(v,H_0)$ is at most the number edge-updates incident upon v.

□

Corollary 7.46. *At least* $\Omega(\varepsilon\gamma\cdot(L_dL^2)^{-1}\cdot(d/2^j))$ *edge-updates in the epoch* $\left[\tau^0,\tau^1\right]$ *are incident upon v.*

Proof. Note that $e^{\gamma} \in (1,e)$ since $\gamma \in (0,1)$. The corollary now follows from Corollary 7.44 and Claim 7.45. □

Part III

DYNAMIC ALGORITHM FOR BIPARTITE GRAPHS: FULL DETAILS

8 Notations and Preliminaries

This part of the writeup presents our dynamic algorithm for bipartite graphs for maintaining a better than 2 approximation to the size of the maximum matching. The main result is summarized in Theorem 8.2.

Notations. We now define some notations that will be used throughout this half of the paper. Let $G = (V,E)$ denote the input graph with $n = |V|$ nodes and $m = |E|$ edges. Given any subset of edges $E' \subseteq E$ and any node $v \in V$, we let $N_v(E') = \{u \in V : (u,v) \in E'\}$ denote the set of neighbors of v that are connected to v via an edge in E'. Furthermore, we let $\deg_v(E') = |N_v(E')|$ denote the number of edges in E' that are incident upon v. Finally, for any subset of edges $E' \subseteq E$, we let $V(E') = \{v \in V : \deg_v(E') > 0\}$ denote the set of endpoints of the edges in E'.

Fractional assignments. A "fractional assignment" is a function $w : E \to \mathbf{R}^+$. It assigns a nonnegative "weight" $w(e)$ to every edge $e \in E$ in the input graph $G = (V,E)$. The set of edges $\text{Support}(w) = \{e \in E : w(e) > 0\}$ with positive weights is called the "support" of the fractional assignment w. The "weight" of a node $v \in V$ under w is given by $W_v(w) = \sum_{(u,v)\in E} w(u,v)$. In other words, $W_v(w)$ denotes the total weight received by v from its incident edges, under the fractional assignment w. We now define the "addition" of two fractional assignments. Given any two fractional assignments w, w', we say that $w + w'$ is a new fractional assignment such that $(w+w')(e) = w(e) + w'(e)$ for every edge $e \in E$. It is easy to check that this addition operation is commutative and associative, i.e., we have $w + w' = w' + w$ and $w + (w' + w'') = (w + w') + w''$ for any three fractional assignments w, w', w'' defined on the same graph. Given any subset of edges $E' \subseteq E$ and any fractional assignment w, we let $w(E') = \sum_{e\in E'} w(e)$ denote the sum of the weights of the edges in E' under w. The "size" of a fractional assignment w is defined as $w(E) = \sum_{e\in E} w(e)$.

Fractional b-matchings. Suppose that we assign a "capacity" $b_v \geq 0$ to each node $v \in V$ in the input graph $G = (V,E)$. A fractional assignment w is called a "fractional-b-matching" with respect to these capacities iff $W_v(w) \leq b_v$ for every node $v \in V$. The size of this fractional b-matching is given by $w(E) = \sum_{e\in E} w(e)$, and its support is defined as $\text{Support}(w) = \{e \in E : w(e) > 0\}$. We say that a fractional b-matching w is "maximal" iff for every edge $(u,v) \in E$, either $W_v(w) = b_v$ or $W_u(w) = b_u$.

"Extending" a fractional b-matching. During the course of our algorithm, we will often consider some subgraph $G' = (V', E')$ of the input graph $G = (V,E)$, with $V' \subseteq V$ and $E' \subseteq E$, and define a fractional b-matching $w' : E' \to \mathbf{R}^+$ on G' with respect to the node-capacities $\{b'(v)\}, v \in V'$. In such cases, to ease notation, we will often pretend that w' is a fractional b-matching on G itself. We will do this by setting $w'(e) = 0$ for all edges $e \in E \setminus E'$ and $b'(v) = 0$ for all nodes $v \in V \setminus V'$. With this notational convention, we will be able to "add" two fractional assignments w', w'' defined on two different subgraphs G', G'' of G.

Fractional matchings. A fractional assignment w in the input graph $G = (V,E)$ is called a "fractional matching" iff we have $W_v(w) \in [0,1]$ for all nodes $v \in V$. In other words, this is a fractional b-matching where every node has capacity one. The "size" of this fractional matching is given by $w(E) = \sum_{e\in E} w(e)$, and its support is defined as $\text{Support}(w) = \{e \in E : w(e) > 0\}$. We say that a fractional matching w is "maximal" iff for every edge $(u,v) \in E$, either $W_v(w) = 1$ or $W_u(w) = 1$.

(Integral) matchings. In the input graph $G = (V,E)$, an (integral) matching $M \subseteq E$ is a subset of edges that have no common endpoints. This can be thought of as a fractional matching $w_M : E \to [0,1]$ with $\text{Support}(w_M) = M$ such that $w_M(e) = 1$ for all edges $e \in M$. The size of this matching is given by $|M| = w_M(E)$. We say that the matching M is "maximal" iff for every edge $(u,v) \in E \setminus M$, at least one of its endpoints $\{u,v\}$ is matched under M. This is equivalent to the condition that the corresponding fractional matching w_M is maximal. We will need the following theorem, which shows that the maximum size of a fractional matching is no more than the maximum size of an integral matching in a bipartite graph.

Theorem 8.1. *In an input graph $G = (V,E)$, consider any fractional matching w with support $E' \subseteq E$. If the graph G is bipartite, then there is a matching $M \subseteq E'$ of size at least $w(E)$.*

Our result. In Section 9, we first fix three parameters ε, δ, K as per equations 163 — 169 (we assume that the number of nodes n in the input graph is sufficiently large). Then we describe some invariants that define three fractional assignments w, w^r, w_1^* in $G = (V, E)$. In Theorem 9.4, we show that $(w + w^r + w_1^*)$ forms a fractional matching in $G = (V, E)$. In Theorem 9.5, we show that the size of $(w + w^r + w_1^*)$ is a $(1/f)$-approximation to the maximum possible size of a fractional matching in $G = (V, E)$, where:

$$f = (1/2) \cdot \left(\frac{(1 + \delta/3)}{(1 + \varepsilon)} - 4K\varepsilon \right).$$

In Section 10, we show how to maintain a $(1+\varepsilon)^2$-approximation to the size of $(w + w^r + w_1^*)$ with $\kappa(n) = O((10/\varepsilon)^{K+8} \cdot n^{2/K})$ amortized update time (see Theorem 10.5). This implies that we can maintain a $(1+\varepsilon)^2/f$-approximation to the size of the maximum fractional matching in G with $\kappa(n)$ update time. By Theorem 8.1, the size of the maximum fractional matching in G equals the size of the maximum cardinality (integral) matching in G. Thus, we reach the following conclusion.

- In $O((10/\varepsilon)^{K+8} \cdot n^{2/K})$ amortized update time, we can maintain a $(1+\varepsilon)^2/f$-approximation to the size of the maximum matching in G.

We now set the values of ε, δ, and define two new parameters α_K, β_K as follows.

$$\alpha_K = \frac{2 \cdot (1+\varepsilon)^3}{(1 + \delta/3) - 4K\varepsilon(1+\varepsilon)} \text{ and } \beta_K = (10/\varepsilon)^{K+8}, \text{ where } \varepsilon = \frac{1}{36K \cdot 10^{K+4}} \text{ and } \delta = 10^4 \cdot K\varepsilon. \quad (154)$$

It is easy to check that this setting of values satisfies equations 163 – 169. Further, note that $\alpha_K = (1+\varepsilon)^2/f$. Thus, we get the following theorem:

Theorem 8.2. *Fix any positive integer K and define α_K, β_K as per equation 154. In a dynamic setting, we can maintain a α_K-approximation to the value of the maximum matching in a bipartite graph $G = (V, E)$ with $O(\beta_K \cdot n^{2/K})$ amortized update time. Note that $1 \leq \alpha_K < 2$ for every sufficiently large integer K.*

8.1 An important technical theorem

We devote this section to the proof of the following theorem. This result will be crucially used later on in the design and analysis of our dynamic algorithm.

Theorem 8.3. *Consider a bipartite graph $G = (V, E)$, where the node-set V is partitioned into two subsets $A \subseteq V$ and $B = V \setminus A$ such that every edge $e \in E$ has one endpoint in A and another endpoint in B. Fix any number $\lambda \in [0, 1/2]$, and suppose that each node $v \in B$ has a capacity $b(v) = 2\lambda$. Furthermore, suppose that each node $u \in A$ has a capacity $b(u) \in [0, \lambda]$. Let w be a maximal fractional b-matching in the graph G with respect to these capacities. Thus, for every edge $(u, v) \in E$, we have either $W_u(w) = b(u)$ or $W_v(w) = b(v)$. Further, for every node $v \in V$, we have $0 \leq W_v(w) \leq b(v)$.*

Let $M \subseteq E$ be a matching in G, i.e., no two edges in M share a common endpoint. Finally, let $A(M) = \{u \in A : deg_M(u) = 1\}$ and $B(M) = \{v \in B : deg_M(v) = 1\}$ respectively denote the set of nodes from A and B that are matched under M. Then we have:

$$\sum_{v \in V} W_v(w) \geq (4/3) \cdot \sum_{u \in A(M)} b(u).$$

Proof of Theorem 8.3.

For each node $u \in A(M)$, define $\Delta_u(w) = b(u) - W_u(w)$ to be the "slack" at node u with respect to w. Since for all nodes $u \in A(M)$, we have $0 \leq W_u(w) \leq b(u) \leq \lambda$, we infer that $\Delta_u(w) \in [0, \lambda]$ for all nodes $u \in A(M)$. We will show that:

$$\sum_{u \in A(M)} \Delta_u(w) \leq (1/2) \cdot \sum_{u \in A} W_u(w) \quad (155)$$

We claim that equation 155 implies Theorem 8.3. To see why this is true, consider two possible cases.

- *Case 1.* $\sum_{u\in A(M)} \Delta_u(w) \leq (1/3)\cdot\sum_{u\in A(M)} b(u)$.
  In this case, we have:

$$\begin{aligned}\sum_{u\in A(M)} W_u(w) &= \sum_{u\in A(M)} (b(u)-\Delta_u(w)) \\ &= \sum_{u\in A(M)} b(u) - \sum_{u\in A(M)} \Delta_u(w) \\ &\geq \sum_{u\in A(M)} b(u) - (1/3)\cdot\sum_{u\in A(M)} b(u) \\ &= (2/3)\cdot\sum_{u\in A(M)} b(u) \end{aligned} \tag{156}$$

  Now, since an edge $e\in E$ contributes the same amount $w(e)$ to each of the sums $\sum_{u\in A} W_u(w)$ and $\sum_{v\in B} W_v(w)$, we have $\sum_{u\in A} W_u(w) = \sum_{v\in B} W_v(w)$. Thus, we get:

$$\begin{aligned}\sum_{v\in V} W_v(w) &= 2\cdot\sum_{u\in A} W_u(w) \\ &\geq 2\cdot\sum_{u\in A(M)} W_u(w) &(157)\\ &\geq (4/3)\cdot\sum_{u\in A(M)} b(u) &(158)\end{aligned}$$

  Equation 157 holds since $A(M)\subseteq A$. Equation 158 follows from equation 156. The theorem follows from equation 158.
- *Case 2.* $\sum_{u\in A(M)} \Delta_u(w) > (1/3)\cdot\sum_{u\in A(M)} b(u)$.
  In this case, we have:

$$\begin{aligned}\sum_{u\in A} W_u(w) &\geq 2\cdot\sum_{u\in A(M)} \Delta_u(w) &(159)\\ &> (2/3)\cdot\sum_{u\in A(M)} b(u) &(160)\end{aligned}$$

  Equation 159 follows from equation 155. Next, just as in Case 1, we argue that an edge $e\in E$ contributes the same amount $w(e)$ to each of the sums $\sum_{u\in A} W_u(w)$ and $\sum_{v\in B} W_v(w)$. So we have: $\sum_{u\in A} W_u(w) = \sum_{v\in B} W_v(w)$. Thus, we get:

$$\begin{aligned}\sum_{v\in V} W_v(w) &= 2\cdot\sum_{u\in A} W_u(w) \\ &> (4/3)\cdot\sum_{u\in A(M)} b(u) &(161)\end{aligned}$$

  Equation 161 follows from equation 160. The theorem follows from equation 161.

Thus, in order to prove the theorem, it suffices to prove equation 155. This is shown below.

- *Proof of equation 155.*
  Let $A^*(M) = \{u\in A(M) : \Delta_u(w) > 0\}$ denote the subset of nodes in $A(M)$ that have nonzero slack under w. For each node $u\in A(M)$, let $u(M)\in B(M)$ denote the node u is matched to under M. Since w is a maximal fractional b-matching with respect to the capacities $\{b(v)\}, v\in V$, we infer that $W_{u(M)}(w) = b(u(M)) = 2\lambda$ for all nodes $u\in A^*(M)$. Further, we note that $\Delta_u(w) = b(u) - W_u(w) \leq b(u) \leq \lambda$ for all nodes $u\in A^*(M)\subseteq A$. Thus, we get:

$$\Delta_u(w) \leq (1/2)\cdot W_{u(M)}(w) \text{ for all nodes } u\in A^*(M). \tag{162}$$

From equation 162, we infer that:

$$\begin{aligned}\sum_{u\in A(M)}\Delta_u(w) &= \sum_{u\in A(M)\setminus A^*(M)}\Delta_u(w)+\sum_{u\in A^*(M)}\Delta_u(w)\\ &= \sum_{u\in A^*(M)}\Delta_u(w)\\ &\leq (1/2)\cdot\sum_{u\in A^*(M)}W_{u(M)}(w)\\ &\leq (1/2)\cdot\sum_{v\in B}W_v(w)\\ &= (1/2)\cdot\sum_{u\in A}W_u(w)\end{aligned}$$

The last equality holds since each edge $e\in E$ contributes the same amount $w(e)$ towards the sums $\sum_{v\in B}W_v(w)$ and $\sum_{u\in A}W_u(w)$, and so these two sums are equal. This concludes the proof of equation 155.

9 Invariants maintained by our algorithm

Throughout the rest of this half of the paper, we fix a sufficiently large integral constant $K\geq 10$, and sufficiently small constants $\varepsilon,\delta\in(0,1)$. We will assume that K,ε,δ satisfy the following guarantees.

$$\varepsilon = 1/N \text{ for some positive integer } N\gg K, \text{ and } \delta < \frac{1}{36\cdot 10^{K-2}}. \tag{163}$$

$$\varepsilon \ll \delta, \text{ and } \delta \text{ is an integral multiple of } \varepsilon. \tag{164}$$

$$\delta < \frac{1}{13\cdot 10^{K}} \tag{165}$$

$$\varepsilon \gg 1/n^{1/K}, \text{ where } n=|V| \text{ is the number of nodes in the input graph.} \tag{166}$$

$$K < n^{1/K} \tag{167}$$

$$\sum_{j=0}^{i} n^{j/K} \leq 2\cdot n^{i/K} \text{ for all } i\in\{1,\ldots,K\}. \tag{168}$$

$$\log n \leq n^{1/K}. \tag{169}$$

We maintain a family of K subgraphs $G_1,\ldots,G_K$ of the input graph $G=(V,E)$. For $1\leq i\leq K$, let Z_i and E_i respectively denote the node-set and the edge-set of the i^{th} subgraph, i.e., $G_i=(Z_i,E_i)$. We ensure that:

(a) $V=Z_K\supseteq Z_{K-1}\supseteq\cdots\supseteq Z_1$.

(b) For $1\leq i\leq K$, the set $E_i=\{(u,v)\in E : u,v\in Z_i\}$ consists of the edges with both endpoints in Z_i.

We define the "level" of a node $v\in V$ to be the minimum index $i\in\{1,\ldots,K\}$ such that $v\in Z_i$. We denote the level of a node v by $\ell(v)$. Thus, we have:

$$\ell(v)=\min_{i\in[1,K]}\{v\in Z_i\} \quad \text{for all nodes } v\in V. \tag{170}$$

The corollary below follows from our definition of the level of a node.

Corollary 9.1. *Consider any node $v\in V$ with level $\ell(v)=i\in\{1,\ldots,K\}$. We have:*

$$v\in Z_j \text{ for all } j\in\{i,\ldots,K\}, \text{ and } v\notin Z_j \text{ for all } j\in\{1,\ldots,i-1\}.$$

9.1 An overview of the structures maintained by our algorithm

For each index $i \in \{2,\ldots,K\}$, our algorithm will maintain the following structures.

1. The subgraph $G_i = (Z_i, E_i)$.

2. For each node $v \in Z_i$, two capacities $b_i(v), b_i^r(v) \geq 0$. The former is called the "normal" capacity of the node at level i, whereas the latter is called the "residual" capacity of the node at level i.

3. A fractional b-matching w_i in the subgraph $G_i = (Z_i, E_i)$ with respect to the normal capacities $\{b_i(v)\}$, $v \in Z_i$. This is called the "normal" fractional assignment at level i. The support of w_i will be denoted by $H_i = \{e \in E_i : w_i(e) > 0\}$. The fractional assignment w_i will be "uniform", in the sense that every edge $e \in H_i$ gets the same weight $w_i(e) = 1/d_i$, where $d_i = n^{(i-1)/K}$.

   To ease notation, we will often extend w_i to the input graph $G = (V, E)$ as described in Section 8. Thus, $b_i(v) = 0$ for all $v \in V \setminus Z_i$ and $w_i(e) = 0$ for all $e \in E \setminus E_i$. Accordingly, if a node v does not belong to Z_i, then $W_v(w_i) = 0$.

4. For each node $v \in Z_i$, a "discretized node-weight" $\mathscr{W}_v(w_i) \leq b_i(v)$ that is an integral multiple of ε and gives a close estimate of the normal node-weight $W_v(w_i) = \sum_{(u,v)\in E_i} w_i(u,v)$. Intuitively, one can think of $\mathscr{W}_v(w_i)$ as follows: It "rounds up" the value of $W_v(w_i)$ to a multiple of ε, while ensuring that the rounded value (1) does not exceed $b_i(v)$, and (2) does not differ too much from the actual value. The second property will be used in the analysis of our algorithm.

   We will often extend this notation to the entire node-set V by assuming that $\mathscr{W}_v(w_i) = 0$ for all nodes $v \in V \setminus Z_i$. We will also set $\mathscr{W}_v(w) = \sum_{j=2}^{K} \mathscr{W}_v(w_j)$ for all nodes $v \in V$ (see equation 171 below).

5. A fractional b-matching w_i^r in the subgraph $G_i = (Z_i, E_i)$ with respect to the residual capacities $\{b_i^r(v)\}, v \in Z_i$. This is called the "residual" fractional assignment at level i.

   To ease notation, we will often extend w_i^r to the input graph $G = (V, E)$ as described in Section 8. Thus, $b_i^r(v) = 0$ for all $v \in V \setminus Z_i$ and $w_i^r(e) = 0$ for all $e \in E \setminus E_i$. Accordingly, we also set $W_v(w_i^r) = 0$ for all nodes $v \in V \setminus Z_i$.

6. A partition of the node-set Z_i into three subsets: T_i, B_i, S_i. The nodes in T_i, B_i and S_i will be respectively called "tight", "big" and "small" at level i.

We will often use the phrase "structures for level i" while describing our algorithm. This is defined below.

Definition 9.2. *For every $i \in \{2,\ldots,K\}$, the phrase "structures for level i" refers to the subgraph $G_i = (Z_i, E_i)$, the node-capacities $\{b_i(v), b_i^r(v)\}$, $v \in Z_i$, the fractional assignments w_i, w_i^r, the discretized node-weights $\{\mathscr{W}_v(w_i)\}, v \in Z_i$, and the partition of the node-set Z_i into subsets T_i, B_i, S_i.*

Next, our algorithm will maintain the following structures for level one.

1. The subgraph $G_1 = (Z_1, E_1)$.

2. For each node $v \in Z_1$, a capacity $b_1^*(v) \geq 0$.

3. A fractional b-matching w_1^* in $G_1 = (Z_1, E_1)$ with respect to the node-capacities $\{b_1^*(v)\}, v \in Z_1$. To ease notation, we will often extend w_1^* to the input graph $G = (V, E)$ as described in Section 8. So we will set $b_1^*(v) = 0$ for all $v \in V \setminus Z_1$ and $w_1^*(e) = 0$ for all $e \in E \setminus E_1$.

Definition 9.3. *The phrase "structures for level 1" will refer to the subgraph $G_1 = (Z_1, E_1)$, the node-capacities $\{b_1^*(v)\}$, $v \in Z_1$, and the fractional assignment w_1^*.*

Our algorithm will ensure the following property.

- Fix any $1 \leq i \leq K$. The structures for level i depend only on the structures for levels $i+1 \leq j \leq K$. Equivalently, the structures for a level $j < i$ do not influence the structures for level i.

We will define two fractional assignments – w and w^r – as follows.

$$w = \sum_{j=2}^{K} w_j \text{ and } w^r = \sum_{j=2}^{K} w_j^r \tag{171}$$

We will show that the fractional assignment $(w + w_1^* + w^r)$ forms a fractional matching in the input graph $G = (V, E)$. Further, the size of this fractional matching is *strictly* within a factor of 2 of the size of the maximum cardinality matching in G. Our main results are thus summarized in the two theorems below. Note that equations 163 and 164 guarantee that $f > 1/2$ (see Theorem 9.5).

Theorem 9.4. *The fractional assignment $(w + w_1^* + w^r)$ is a fractional matching in the graph $G = (V, E)$.*

Theorem 9.5. *The size of the fractional assignment $(w + w_1^* + w^r)$ is at least f times the size of the maximum cardinality matching in the input graph $G = (V, E)$, where*

$$f = (1/2) \cdot \left(\frac{(1 + \delta/3)}{(1 + \varepsilon)} - 4K\varepsilon \right).$$

Organization for the rest of this section.

- In Section 9.2, we describe the invariants that are used to define the structures for levels $j \in \{2, \ldots, K\}$.
- In Section 9.3, we show that the above invariants are "consistent" in the following sense: There is a way to construct the structures for all levels $j > 1$ that satisfy these invariants.
- In Section 9.4, we describe the invariants that are used to define the structures for level one.
- In Section 9.5, we again prove the "consistency" of these invariants. Specifically, we show that there is a way to construct the structures for level one that satisfy the invariants.
- In Section 9.6, we derive some properties from the invariants that will be useful later while analyzing the update time of our algorithm.
- In Section 9.7, we prove Theorem 9.4 by showing that the fractional assignment $(w + w_1^* + w^r)$ forms a fractional matching in the input graph $G = (V, E)$.
- Finally, in Section 9.8, we bound the size of the fractional assignment $(w + w_1^* + w^r)$ and prove the required approximation guarantee as stated in Theorem 9.5.

Before moving on to Section 9.2, we introduce a notation that will be used multiple times. For every $i \in \{1, \ldots, K-1\}$, we define the "small-index" $s_i(v)$ of a node $v \in Z_i$ to be the minimum level $j \in \{i+1, \ldots, K\}$ where the node is small (i.e., $v \in S_j$). If $i = K$ or if $v \notin S_j$ for all $j \in \{i+1, \ldots, K\}$, then we define $s_i(v) = \infty$.

$$s_i(v) = \begin{cases} \infty & \text{if } i = K \text{ or } v \notin S_j \ \ \forall j \in \{i+1, \ldots, K\}; \\ \min\left\{ j \in \{i+1, \ldots, K\} : v \in S_j \right\} & \text{otherwise.} \end{cases} \tag{172}$$

Intuitively, $s_i(v)$ captures the notion of the "most recent" level larger than i where v is small.

9.2 Invariants for levels $i \in \{2,\ldots,K\}$

Fix any level $i \in \{2,\ldots,K\}$, and recall Definition 9.2. Suppose we are given the structures for all the levels $j \in \{i+1,\ldots,K\}$. Given this input, we present the invariants that determine the structures for level i.

First, in Definition 9.6, we specify how to derive the subgraph $G_i = (Z_i, E_i)$. It states that if $i < K$, then the node-set Z_i consists of all the "non-tight" nodes (see Section 9.1) at level $i+1$. Else if $i = K$, then the set Z_i consists of all the nodes in the input graph $G = (V,E)$. Further, the set E_i consists of the edges in G with both endpoints in Z_i. In other words, G_i is the subgraph of G induced by the subset of nodes $Z_i \subseteq V$.

Definition 9.6. *For all $i \in \{2,\ldots,K\}$, the node-set Z_i is defined as follows.*

$$Z_i = \begin{cases} V & \text{for } i = K; \\ Z_{i+1} \setminus T_{i+1} & \text{for } i \in \{1,\ldots,K-1\}. \end{cases}$$

Further, we define $E_i = \{(u,v) \in E : u,v \in Z_i\}$ to be the set of edges in E with both endpoints in Z_i.

In Definition 9.7, we specify how to derive the normal node-capacity of a node $v \in Z_i$. This is determined by two quantities: (1) the total *discretized weight* received by the node from the levels larger than i; and (2) the value of the index $s_i(v)$. From equation 172, it follows that the value of $s_i(v)$ is determined by the partitions of the node-sets $\{Z_j\}$ into subsets $\{T_j, S_j, B_j\}$ for all $i < j \leq K$. Thus, the node-capacities $\{b_i(v)\}, v \in Z_i$, are uniquely determined by the structures for levels $j \in \{i+1,\ldots,K\}$.

Definition 9.7. *Fix any level $i \in \{2,\ldots,K\}$ and any node $v \in Z_i$. The capacity $b_i(v)$ is defined as follows.*

$$b_i(v) = \begin{cases} 1 - \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j) & \text{if } s_i(v) = \infty; \\ \\ 1 - 5 \cdot 10^{(K - s_i(v)+1)} \cdot \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j) & \text{else if } s_i(v) \in \{i+1,\ldots,K\}. \end{cases} \tag{173}$$

Note that by equation 164, δ is an integral multiple of ε. Furthermore, we have $1 = N \cdot \varepsilon$ for some integer N (see equation 163). In Section 9.1, while describing the structures for a level, we stated that the discretized weight of a node is also an integral multiple of ε. Thus, for every node $v \in Z_i$, we infer that $\mathscr{W}_v(w_j)$ is an integral multiple of ε for all levels $j > i$. Hence, Definition 9.7 implies that $b_i(v)$ is an integral multiple of ε for all nodes $v \in Z_i$. In Section 9.3, we show that $b_i(v) > 0$ for every node $v \in Z_i$ (see Lemma 9.15). Accordingly, we get: $b_i(v) \geq \varepsilon$ for every node $v \in Z_i$.

In Definition 9.8 and Invariant 9.9, we describe how to construct the fractional b-matching w_i in $G_i = (Z_i, E_i)$ with respect to the capacities $\{b_i(v)\}$. Every edge in the support of w_i gets a weight $1/d_i$. Further, this weight $1/d_i$ is negligibly small compared to the smallest node-capacity ε (see equation 166). Hence, it is indeed possible to construct a nontrivial fractional matching w_i with nonempty support H_i in this manner.

Definition 9.8. *We define $d_i = n^{(i-1)/K}$ for all $i \in \{1,\ldots,K\}$.*

Invariant 9.9. *Consider any level $2 \leq i \leq K$. We maintain a fractional b-matching in G_i with respect to the node-capacities $\{b_i(v)\}, v \in Z_i$, and denote this fractional b-matching by w_i. Thus, we have: $W_v(w_i) \leq b_i(v)$ for all nodes $v \in Z_i$. We define $H_i = \{e \in E_i : w_i(e) > 0\}$ to be the support of w_i. We ensure that $w_i(e) = 1/d_i$ for all $e \in H_i$. In other words, the fractional assignment w_i is "uniform", in the sense that it assigns the same weight to every edge in its support.*

After constructing the fractional assignment w_i, as per Invariant 9.10 we give each node $v \in Z_i$ a discretized weight $\mathscr{W}_v(w_i)$ that is an integral multiple of ε. These discretized weights serve as estimates of actual node-weights $\{W_v(w_i)\}$. In Section 9.3, we show that it is indeed possible to satisfy Invariant 9.10.

Invariant 9.10. *Consider any level $i \in \{2,\ldots,K\}$. For every node $v \in Z_i$, we maintain a value $0 \leq \mathscr{W}_v(w_i) \leq b_i(v)$ so that $\mathscr{W}_v(w_i) - 2\varepsilon < W_v(w_i) \leq \mathscr{W}_v(w_i)$. Further, $\mathscr{W}_v(w_i)$ is an integral multiple of ε.*

Invariant 9.12 implies that w_i is "maximal" with respect to the discretized node-weights $\{\mathscr{W}_v(w_i)\}$. To be more specific, the node-set Z_i is partitioned into three subsets T_i, B_i and S_i. The nodes in T_i, B_i and S_i are respectively called "tight", "big" and "small" at level i (see Definition 9.11). A node v is tight iff $\mathscr{W}_v(w_i) = b_i(v)$. For a tight node v, if we take an edge (u,v) from $E_i \setminus H_i$ and insert it into H_i, then the weights $W_v(w_i), \mathscr{W}_v(w_i)$ might exceed the capacity $b_i(v)$ – thereby violating Invariant 9.10. But this issue does not arise if the node v is non-tight. Accordingly, the invariant requires every edge in $E_i \setminus H_i$ to have at least one non-tight endpoint.

In Section 9.3, we show that the subsets $S_i, T_i \subseteq Z_i$ are mutually disjoint (see Lemma 9.17). This implies that the node-set Z_i is indeed partitioned into three subsets – T_i, S_i and B_i – as claimed in the invariant below. Intuitively, this claim holds since δ is sufficiently small. To be more specific, consider a node $v \in T_i$. By Definition 9.11, we have $\mathscr{W}_v(w_i) = b_i(v)$ for such a node v. Hence, from Definition 9.7, we infer that either $\mathscr{W}_v(w_i) = b_i(v) = 1 - \delta - \sum_{j>i} \mathscr{W}_v(w_j)$, or $\mathscr{W}_v(w_i) = b_i(v) = 1 - 5 \cdot 10^{K-s_i(v)+1} \cdot \delta - \sum_{j>i} \mathscr{W}_v(w_j)$. Rearranging the terms, we get: either $\sum_{j \geq i} \mathscr{W}_v(w_j) = 1 - \delta$ or $\sum_{j \geq i} \mathscr{W}_v(w_j) = 1 - 5 \cdot 10^{K-s_i(v)+1} \cdot \delta$. In either case, if δ is sufficiently small, then we can infer that $\sum_{j \geq i} \mathscr{W}_v(w_j) > 8 \cdot 10^{K-i} \cdot \delta$, which implies that $v \notin S_i$ by Definition 9.11. In other words, a tight node can never be small, which guarantees that $S_i \cap T_i = \emptyset$.

Definition 9.11. *At each level $2 \leq i \leq K$, the node-sets $T_i, B_i, S_i \subseteq Z_i$ are defined as follows.*

$$T_i = \{v \in Z_i : \mathscr{W}_v(w_i) = b_i(v)\} \tag{174}$$

$$S_i = \left\{ v \in Z_i : \sum_{j=i}^{K} \mathscr{W}_v(w_j) \leq 8 \cdot 10^{K-i} \cdot \delta \right\} \tag{175}$$

$$B_i = Z_i \setminus (T_i \cup S_i) \tag{176}$$

Invariant 9.12. *At each level $i \in \{2,\ldots,K\}$, the node-set Z_i is partitioned by the subsets T_i, B_i, S_i. Further, for every edge $(u,v) \in E_i$ with $u,v \in Z_i \setminus T_i$, we have $(u,v) \in H_i$. In other words, every edge between two non-tight nodes belongs to the support of w_i.*

The next definition specifies how to derive the residual capacity of a node $v \in Z_i$ based on (a) the partition of the node-set Z_i into subsets T_i, S_i, B_i and (b) the value of $s_i(v)$. From equation 172, it follows that the value of $s_i(v)$, in turn, depends on the partitions of $\{Z_j\}$ into subsets $\{T_j, S_j, B_j\}$ for all $j > i$.

Definition 9.13. *Fix any level $i \in \{2,\ldots,K\}$ and any node $v \in Z_i$. The capacity $b_i^r(v)$ is defined as follows.*

$$b_i^r(v) = \begin{cases} 0 & \text{if } v \notin (S_i \cup T_i); \\ 8 \cdot 10^{(K-i)} \cdot \delta & \text{else if } v \in S_i; \\ \delta & \text{else if } v \in T_i \text{ and } s_i(v) = \infty; \\ 4 \cdot 10^{(K-s_i(v)+1)} \cdot \delta & \text{else if } v \in T_i \text{ and } s_i(v) \in \{i+1,\ldots,K\}. \end{cases} \tag{177}$$

Note that $b_i^r(v) \geq 0$ for all nodes $v \in Z_i$. Further, note that $b_i^r(v)$ is nonzero only if $v \in T_i \cup S_i$. We define a subgraph $G_i^r = (T_i \cup S_i, E_i^r)$ of G_i where the edge-set $E_i^r = \{(u,v) \in E_i : u \in T_i, v \in S_i\}$ consists of those edges in E_i that have one tight and one small endpoints. The edges in E_i^r are called "residual edges in level i". Similarly, the subgraph G_i^r is called the "residual subgraph" in level i. The next invariant states how to construct the residual fractional matching w_i^r on the residual subgraph G_i^r.

Invariant 9.14. *For all $i \in [2,K]$, let $E_i^r = \{(u,v) \in E_i : u \in T_i, v \in S_i\}$ be the set of edges in E_i with one tight and one small endpoints. Let $G_i^r = (T_i \cup S_i, E_i^r)$ be the subgraph of $G_i = (Z_i, E_i)$ induced by these edges. The*

fractional assignment w_i^r is a maximal fractional b-matching in G_i^r with respect to the residual capacities $\{b_i^r(v)\}, v \in T_i \cup S_i$. Thus, for every edge $(u,v) \in E^r$, either $W_v(w_i^r) = b_i^r(v)$ or $W_u(w_i^r) = b_i^r(u)$. We refer to E_i^r, G_i^r and w_i^r as residual edges, residual subgraph and residual fractional matching at level i respectively. We implicitly assume that $w_i^r(e) = 0$ for all edges $e \in E_i \setminus E_i^r$. This ensures that w_i^r is also a valid fractional b-matching in G_i with respect to the residual capacities described in Definition 9.13.

We now explain the link between Invariant 9.14 and Theorem 8.3. Suppose that $\lambda = 4 \cdot 10^{(K-i)} \cdot \delta$. Recall Definition 9.13. Consider any node $v \in T_i$. If $s_i(v) = \infty$, then we have $b_i^r(v) = \delta \leq \lambda$. On the other hand, if $s_i(v) \in \{i+1, \ldots, K\}$, then we also have $b_i^r(v) = 4 \cdot 10^{(K-s_i(v)+1)} \cdot \delta \leq 4 \cdot 10^{(K-i)} \cdot \delta \leq \lambda$. Thus, we have $b_i^r(v) \in [0, \lambda]$ for all tight nodes $v \in T_i$. In contrast, we have $b_i^r(v) = 8 \cdot 10^{(K-i)} \cdot \delta = 2\lambda$ for all small nodes $v \in S_i$. We therefore reach the following conclusion: If we set $\lambda = 4 \cdot 10^{(K-i)} \cdot \delta$, $A = T_i$, $B = S_i$, $E = E_i^r$ and $G = G_i^r$, then we can apply Theorem 8.3 on the residual fractional matching w_i^r.

9.3 Feasibility of the structures for levels $\{2, \ldots, K\}$

Recall Definition 9.2. We will show that there is a way to satisfy all the invariants described so far. In other words, we will show that we can build the structures for levels $\{2, \ldots, K\}$ in a way that does not violate any invariant. We will prove this by induction. For the rest of this section, fix any $i \in \{2, \ldots, K\}$, and suppose that we have already built the structures for levels $\{i+1, \ldots, K\}$ without violating any invariant. It remains show how to build the structures for level i. Towards this end, we first construct the subgraph $G_i = (Z_i, E_i)$.

- Following Definition 9.6, we set $Z_i = Z_{i+1} \setminus T_{i+1}$ if $i < K$. Otherwise, we set $Z_i = Z_K = V$.
- We define $E_i = \{(u,v) \in E : u, v \in Z_i\}$ to be the set of edges in E with endpoints in Z_i.

Next, we define the node-capacities $\{b_i(v)\}, v \in Z_i$ as per Definition 9.7. In Lemma 9.15 we show that every node $v \in Z_i$ has a positive capacity $b_i(v) > 0$. The complete proof of the lemma involves some case-analysis and is deferred to Section 9.3.1. The intuition behind the proof, however, can be summarized as follows.

- If $i = K$, then we have $b_i(v) = 1 - \delta > 0$. Else if $i < K$, then $v \in Z_i$, and hence $v \in Z_{i+1} \setminus T_{i+1}$. Thus, Definition 9.11 and Invariant 9.12 (applied to level $i+1$) implies that $\mathscr{W}_v(w_{i+1}) < b_{i+1}(v)$. This positive gap between $b_{i+1}(v)$ and $\mathscr{W}_v(w_{i+1})$ translates into a positive value for $b_i(v)$ at level i.

Lemma 9.15. *For every node $v \in Z_i$, with $i \in \{2, \ldots, K\}$, we have $b_i(v) > 0$.*

Next, we build a fractional b-matching w_i in the subgraph $G_i = (Z_i, E_i)$ with respect to the node-capacities $\{b_i(v)\}, v \in Z_i$. We let $H_i = \{e \in E_i : w_i(e) > 0\}$ denote the support of the fractional assignment w_i. We ensure that w_i is uniform, in the sense that $w_i(e) = 1/d_i = 1/n^{(i-1)/K}$ for all $e \in E_i$. In other words, w_i assigns the same weight to every edge in its support. This satisfies Invariant 9.9.

We now observe a crucial property. Since δ is a multiple of ε (see equation 164), $1 = N \cdot \varepsilon$ for some integer N (see equation 163), and the discretized node-weights $\{\mathscr{W}_v(w_j)\}$ for levels $j \in \{i+1, \ldots, K\}$ are also multiples of ε (apply Invariant 9.10 to levels $j > i$), Definition 9.7 implies that all the node-capacities $\{b_i(v)\}$ are also multiples of ε.

Observation 9.16. *For every node $v \in Z_i$, with $i \in \{2, \ldots, K\}$, the capacity $b_i(v)$ is an integral multiple of ε.*

Now, suppose that for every node $v \in Z_i$, we define $\mathscr{W}_v(w_i) = \lceil W_v(w_i)/\varepsilon \rceil \cdot \varepsilon$ to be the smallest multiple of ε that is no less than $W_v(w_i)$. Then the discretized weights $\{\mathscr{W}_v(w_i)\}$ clearly satisfy: $\mathscr{W}_v(w_i) - 2\varepsilon \leq W_v(w_i) \leq \mathscr{W}_v(w_i)$. Further, since $b_i(v)$ is also an integral multiple of $W_v(w_i)$, we get: $\mathscr{W}_v(w_i) \leq b_i(v)$ for all $v \in Z_i$. We conclude that the discretized weights $\{\mathscr{W}_v(w_i)\}$ at level i satisfy Invariant 9.10.

We now proceed towards verifying the consistency of Invariant 9.12. First, we need to show that the subsets T_i and S_i from Definition 9.11 are mutually disjoint. This is shown in Lemma 9.17. The proof of Lemma 9.17 appears in Section 9.3.2. For an intuition behind the proof, recall the discussion immediately before Definition 9.11.

Lemma 9.17. *For $i \in \{2,\ldots,K\}$, the subsets $T_i, S_i \subseteq Z_i$ are mutually disjoint.*

Next, we ensure that the fractional b-matching w_i is *maximal with respect to the discretized capacities* $\{\mathscr{W}_v(w_i)\}$. To gain a better understanding of this concept, consider an edge $(u,v) \in E_i \setminus H_i$ that receives zero weight under w_i, and suppose that both its endpoints are non-tight (i.e., $u,v \in Z_i \setminus T_i$). By Definition 9.11, this means that $\mathscr{W}_u(w_i) < b_i(u)$ and $\mathscr{W}_v(w_i) < b_i(v)$. Since each of these quantities are integral multiples of ε (see Observation 9.16 and Invariant 9.10), we infer that $W_u(w_i) \leq \mathscr{W}_u(w_i) \leq b_i(u) - \varepsilon$ and $W_v(w_i) \leq \mathscr{W}_v(w_i) \leq b_i(v) - \varepsilon$. Now, what will happen if we insert this edge (u,v) into H_i by setting $w_i(u,v) = 1/d_i$? To answer this question, we make the following observations.

- Since $1/d_i \leq 1/n^{1/K}$ for $i \geq 2$ (see Definiton 9.8) and since every edge in H_i gets a weight of $1/d_i$, equation 166 implies that $W_u(w_i)$ (resp. $W_v(w_i)$) will increase by at most ε. Accordingly, $\mathscr{W}_u(w_i)$ (resp. $\mathscr{W}_v(w_i)$) will increase by at most ε. Hence, we will still have $\mathscr{W}_u(w_i) \leq b_i(u)$ and $\mathscr{W}_v(w_i) \leq b_i(v)$.

In other words, if both the endpoints of an edge in $E_i \setminus H_i$ are non-tight, then we can insert that edge into H_i without violating any of the invariants. But we cannot reach the same conclusion if at least one of the endpoints is tight. Accordingly, the second part of Invariant 9.12 states that every edge in $E_i \setminus H_i$ will have at least one tight endpoint. In this sense, the fractional assignment w_i is "maximal with respect to the discretized node-capacities".

Next, we define the residual node-capacities $\{b_i^r(v)\}, v \in Z_i$ as per Definition 9.13. Clearly, all these residual capacities are nonnegative. Finally, we compute a maximal fractional b-matching w_i^r with respect to these residual node-capacities $\{b_i^r(v)\}$ as per Invariant 9.14. At this stage, we have finished constructing the structures for level i.

9.3.1 Proof of Lemma 9.15

Fix any level $i \in \{2,\ldots,K\}$ and any node $v \in Z_i$. If $i = K$, then equation 172 implies that $s_i(v) = \infty$, and hence Definition 9.7 guarantees that $b_i(v) = 1 - \delta > 0$. Thus, for the rest of the proof, we suppose that $i \in \{2,\ldots,K-1\}$. Further, by induction hypothesis, we assume that:

$$b_j(v) > 0 \text{ for all } j \in \{i+1,\ldots,K\}. \tag{178}$$

We now consider three possible cases, depending on the value of $s_i(v)$.

- *Case 1.* $s_i(v) = \infty$.

  In this case, equation 172 implies that $s_{i+1}(v) = \infty$. Accordingly, by Definition 9.7 we get:

$$b_{i+1}(v) = 1 - \delta - \sum_{j=i+2}^{K} \mathscr{W}_v(w_j) \tag{179}$$

$$b_i(v) = 1 - \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j) \tag{180}$$

  Since $v \in Z_i$, Definition 9.6 states that $v \in Z_{i+1} \setminus T_{i+1}$. Hence, applying Definition 9.11 we get:

$$\mathscr{W}_v(w_{i+1}) < b_{i+1}(v) \tag{181}$$

From equations 179 and 181, we get:

$$\mathscr{W}_v(w_{i+1}) < 1 - \delta - \sum_{j=i+2}^{K} \mathscr{W}_v(w_j)$$

Rearranging the terms in the above inequality, we get:

$$\sum_{j=i+1}^{K} \mathscr{W}_v(w_j) < 1 - \delta \tag{182}$$

From equations 182 and 180, we get:

$$b_i(v) > 0 \tag{183}$$

The lemma follows from equation 183.

- *Case 2.* $s_i(v) \in \{i+2, \ldots, K\}$.

  In this case, equation 172 implies that $s_i(v) = s_{i+1}(v)$. Accordingly, by Definition 9.7 we get:

$$b_{i+1}(v) = 1 - 5 \cdot 10^{(K - s_i(v) + 1)} \cdot \delta - \sum_{j=i+2}^{K} \mathscr{W}_v(w_j) \tag{184}$$

$$b_i(v) = 1 - 5 \cdot 10^{(K - s_i(v) + 1)} \cdot \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j) \tag{185}$$

  Since $v \in Z_i$, Definition 9.6 states that $v \in Z_{i+1} \setminus T_{i+1}$. Hence, applying Definition 9.11 we get:

$$\mathscr{W}_v(w_{i+1}) < b_{i+1}(v) \tag{186}$$

  From equations 184 and 186, we get:

$$\mathscr{W}_v(w_{i+1}) < 1 - 5 \cdot 10^{(K - s_i(v) + 1)} \cdot \delta - \sum_{j=i+2}^{K} \mathscr{W}_v(w_j)$$

  Rearranging the terms in the above inequality, we get:

$$\sum_{j=i+1}^{K} \mathscr{W}_v(w_j) < 1 - 5 \cdot 10^{(K - s_i(v) + 1)} \cdot \delta \tag{187}$$

  From equations 185 and 187, we get:

$$b_i(v) > 0 \tag{188}$$

  The lemma follows from equation 188.

- *Case 3.* $s_i(v) = i + 1$.

  In this case, applying Definition 9.7 we get:

$$b_i(v) = 1 - 5 \cdot 10^{(K - i)} \cdot \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j) \tag{189}$$

Since $s_i(v) = i+1$, equation 172 implies that $v \in S_{i+1}$. Hence, applying Definition 9.11 we get:

$$\sum_{j=i+1}^{K} \mathscr{W}_v(w_j) \leq 8 \cdot 10^{(K-i-1)} \cdot \delta \tag{190}$$

From equations 189 and 190 we get:

$$\begin{aligned} b_i(v) &\geq 1 - 5 \cdot 10^{(K-i)} \cdot \delta - 8 \cdot 10^{(K-i-1)} \cdot \delta \\ &\geq 1 - 5 \cdot 10^{K} \cdot \delta - 8 \cdot 10^{K} \cdot \delta \\ &= 1 - 13 \cdot 10^{K} \cdot \delta \\ &> 0 \end{aligned} \tag{191}$$

Equation 191 follows from equation 165. The lemma follows from equation 191.

9.3.2 Proof of Lemma 9.17

We first describe a small technical claim.

Claim 9.18. *Consider any level $2 \leq i \leq K$ and any node $v \in Z_i$. We have:*

$$b_i(v) \geq 1 - 5 \cdot 10^{(K-2)} \cdot \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j).$$

Proof. Follows from Definition 9.7 and the fact that $i \geq 2$. □

We now proceed with the proof of Lemma 9.17. For the sake of contradiction, suppose that there is a node $v \in T_i \cap S_i$. Since the node v belongs to T_i, Definition 9.11 and Claim 9.18 imply that:

$$\begin{aligned} \mathscr{W}_v(w_i) &= b_i(v) \\ &\geq 1 - 5 \cdot 10^{(K-2)} \cdot \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j) \end{aligned}$$

Rearranging the terms in the above inequality, we get:

$$\sum_{j=i}^{K} \mathscr{W}_v(w_j) \geq 1 - 5 \cdot 10^{(K-2)} \cdot \delta \tag{192}$$

On the other hand, since the node v belongs to S_i and since $2 \leq i \leq K$, Definition 9.11 implies that:

$$\sum_{j=i}^{K} \mathscr{W}_v(w_j) \leq 8 \cdot 10^{K-2} \cdot \delta \tag{193}$$

From equations 192 and 193 we derive that:

$$8 \cdot 10^{K-2} \cdot \delta \geq 1 - 5 \cdot 10^{K-2} \cdot \delta$$

Rearranging the terms in the above inequality, we get:

$$13 \cdot 10^{K-2} \cdot \delta \geq 1 \tag{194}$$

But equation 194 cannot be true as long as equation 163 holds. This leads to a contradiction. Thus, the sets T_i and S_i have to be mutually disjoint. This concludes the proof of the lemma.

9.4 Invariants for level $i = 1$

Recall Definitions 9.2 and 9.3. Suppose that we are given the structures for all the levels $j \in \{2,\ldots,K\}$. Given this input, in this section we show how to derive the structures for level one. The definition below specifies the subgraph $G_1 = (Z_1, E_1)$. Note that this is exactly analogous to Definition 9.6.

Definition 9.19. *We define $Z_1 = Z_2 \setminus T_2$. Further, the set $E_1 = \{(u,v) \in E : u,v \in Z_1\}$ consists of all the edges with both endpoints in Z_1.*

The definition below shows how to derive the capacity $b_1^*(v)$ of any node $v \in Z_1$.

Definition 9.20. *Consider any node $v \in Z_1$. The capacity $b_1^*(v)$ is defined as follows.*

$$b_1^*(v) = \begin{cases} 1 - \sum_{j=2}^{K} \mathscr{W}_v(w_j) & \text{if } s_1(v) = \infty; \\ \\ 1 - 10^{(K-s_1(v)+1)} \cdot \delta - \sum_{j=2}^{K} \mathscr{W}_v(w_j) & \text{else if } s_1(v) \in \{2,\ldots,K\}. \end{cases} \tag{195}$$

In Section 9.6, we show that $b_1^*(v) > 0$ for every node $v \in Z_1$ (see Lemma 9.22). We now specify how to derive the fractional assignment w_1^*.

Invariant 9.21. *The fractional assignment w_1^* forms a fractional b-matching in $G_1 = (Z_1, E_1)$ with respect to the node-capacities $\{b_1^*(v)\}, v \in Z_1$. Further, the size of w_1^* is at least $1/(1+\varepsilon)$-times the size of every other fractional b-matching in G_1 with respect to the same node-capacities.*

9.5 Feasibility of the structures for level one

The next lemma states that if a node v participates in the subgraph $G_1 = (Z_1, E_1)$, then $b_1^*(v) > 0$. The proof of Lemma 9.22 appears in Section 9.5.1. The intuition behind the proof is similar to the one for Lemma 9.15.

Lemma 9.22. *For every every node $v \in Z_1$, we have $b_i^*(v) > 0$.*

9.5.1 Proof of Lemma 9.22

Fix any node $v \in Z_1$. By Lemma 9.15 we have

$$b_i(v) > 0 \text{ for all } i \in \{2,\ldots,K\}. \tag{196}$$

We consider three possible cases, depending on the value of $s_1(v)$.

- *Case 1.* $s_1(v) = \infty$.

  In this case, equation 172 implies that $s_2(v) = \infty$. Hence, Definitions 9.7 and 9.20 imply that:

$$b_2(v) = 1 - \delta - \sum_{j=3}^{K} \mathscr{W}_v(w_j) \tag{197}$$

$$b_1^*(v) = 1 - \sum_{j=2}^{K} \mathscr{W}_v(w_j) \tag{198}$$

  Since $v \in Z_1$, Definition 9.19 implies that $v \in Z_2 \setminus T_2$. Hence, applying Definition 9.11 we get:

$$\mathscr{W}_v(w_2) < b_2(v) \tag{199}$$

From equations 197 and 199 we get:

$$\mathscr{W}_v(w_2) < 1 - \delta - \sum_{j=3}^{K} \mathscr{W}_v(w_j) \tag{200}$$

Rearranging the terms in the above inequality, we get:

$$\sum_{j=2}^{K} \mathscr{W}_v(w_j) < 1 - \delta \tag{201}$$

From equations 198 and 201 we infer that:

$$b_1^*(v) > 0 \tag{202}$$

The lemma follows from equation 202.

- *Case 2.* $s_1(v) \in \{3, \ldots, K\}$.

  In this case, equation 172 implies that $s_1(v) = s_2(v)$. Hence, applying Definitions 9.7 and 9.20 we get:

$$b_2(v) = 1 - 5 \cdot 10^{(K - s_1(v) + 1)} \cdot \delta - \sum_{j=3}^{K} \mathscr{W}_v(w_j) \tag{203}$$

$$b_1^*(v) = 1 - 10^{(K - s_1(v) + 1)} \cdot \delta - \sum_{j=2}^{K} \mathscr{W}_v(w_j) \tag{204}$$

  Since $v \in Z_1$, Definition 9.19 implies that $v \in Z_2 \setminus T_2$. Hence, applying Definition 9.11 we get:

$$\mathscr{W}_v(w_2) < b_2(v) \tag{205}$$

  From equations 203 and 205 we get:

$$\mathscr{W}_v(w_2) < 1 - 5 \cdot 10^{(K - s_1(v) + 1)} \cdot \delta - \sum_{j=3}^{K} \mathscr{W}_v(w_j)$$

  Rearranging the terms in the above inequality, we get:

$$\sum_{j=2}^{K} \mathscr{W}_v(w_j) < 1 - 5 \cdot 10^{(K - s_1(v) + 1)} \cdot \delta \tag{206}$$

  From equations 204 and 206 we get:

$$b_1^*(v) > 4 \cdot 10^{(K - s_1(v) + 1)} \cdot \delta > 0 \tag{207}$$

  The lemma follows from equation 207.

- *Case 3.* $s_1(v) = 2$.

  In this case, applying Definition 9.20 we get:

$$b_1^*(v) = 1 - 10^{(K-1)} \cdot \delta - \sum_{j=2}^{K} \mathscr{W}_v(w_j) \tag{208}$$

Since $s_1(v) = 2$, equation 172 implies that $v \in S_2$. Hence, applying Definition 9.11 we get:

$$\sum_{j=2}^{K} \mathscr{W}_v(w_j) \leq 8 \cdot 10^{(K-2)} \cdot \delta \tag{209}$$

From equations 208 and 209 we get:

$$\begin{aligned} b_1^*(v) &\geq 1 - 10^{(K-1)} \cdot \delta - 8 \cdot 10^{(K-2)} \cdot \delta \\ &= 1 - 18 \cdot 10^{(K-2)} \cdot \delta \\ &> 0 \end{aligned} \tag{210}$$

Equation 210 follows from equation 163.

9.6 Some useful properties

In this section, we derive some properties from the invariants. These properties will be used later on to analyze the amortized update time of our algorithm. We start by noting that both $1, \delta$ are integral multiples of ε (see equation 163, 164), and that the discretized node-weights are also multiples of ε (see Invariant 9.10). Hence, Definitions 9.7, 9.13, 9.20 imply that all the node-capacities are also multiples of ε.

Observation 9.23. *For every level $i \in \{2, \ldots, K\}$ and every node $v \in Z_i$, both $b_i(v)$ and $b_i^r(v)$ are integral multiples of ε. Further, for every node $v \in Z_1$, the node-capacity $b_1^*(v)$ is an integral multiple of ε.*

From Lemmas 9.15, 9.22 and Observation 9.23, we get the following observation.

Observation 9.24. *For every level $i \in \{2, \ldots, K\}$ and every node $x \in Z_i$, we have $b_i(x) \geq \varepsilon$. Further, for every node $x \in Z_1$, we have $b_1^*(x) \geq \varepsilon$.*

The next observation follows immediately from Definition 9.6.

Observation 9.25. *Consider any level $\ell(v) = i \in \{2, \ldots, K\}$. Then we have $T_i = \{v \in V : \ell(v) = i\}$.*

Lemma 9.26 states that the edges in the subgraph G_{i-1} is drawn from the support of the normal fractional assignment w_i in the previous level i. This holds since only the non-tight nodes at level i get demoted to level $(i-1)$, and all the edges connecting two non-tight nodes are included in the support of w_i.

Lemma 9.26. *For every level $i \in \{2, \ldots, K\}$, we have $E_{i-1} \subseteq H_i$.*

Proof. Consider any edge $(u, v) \in E_{i-1}$ in the subgraph $G_{i-1} = (Z_{i-1}, E_{i-1})$. By Definitions 9.6 and 9.19, we have $u, v \in Z_{i-1} = Z_i \setminus T_i$. Since both $u, v \in Z_i \setminus T_i$, and since E_i is the subset of edges in G induced by the node-set Z_i, we have $(u, v) \in E_i$. Accordingly, Invariant 9.12 implies that $(u, v) \in H_i$. To summarize, every edge $(u, v) \in E_{i-1}$ belongs to the set H_i. Thus, we have $E_{i-1} \subseteq H_i$. □

The next lemma upper bounds the number of edges incident upon a node that can get nonzero weight under a normal fractional assignment w_i. This holds since a normal fractional assignment is *uniform*, in the sense that it assigns the same weight $1/d_i$ to every edge in its support. Hence, not too many edges incident upon a node can be in the support of w_i, for otherwise the total weight of that node will exceed one.

Lemma 9.27. *For each level $i \in \{2, \ldots, K\}$ and every node $v \in Z_i$, we have $deg_v(H_i) \leq d_i$.*

Proof. Invariant 9.9 states that each edge $e \in H_i$ gets a weight of $w_i(e) = 1/d_i$ under the fractional assignment w_i. Thus, for every node $v \in Z_i$, it follows that $W_v(w_i) = (1/d_i) \cdot \deg_v(H_i)$. For the sake of contradiction,

suppose that the lemma is false. Then there must be a node $u \in Z_i$ with $\deg_u(H_i) > d_i$. In that case, we would have:

$$\mathscr{W}_u(w_i) \geq W_u(w_i) = (1/d_i) \cdot \deg_u(H_i) > 1 \geq b_i(u).$$

The first inequality follows from Invariant 9.10. The last inequality follows from Definition 9.7. Thus, we get: $\mathscr{W}_u(w_i) > b_i(u)$, which contradicts Invariant 9.10. Thus, our assumption was wrong, and this concludes the proof of the lemma. □

Finally, we upper bound the maximum degree of a node in any subgraph $G_i = (Z_i, E_i)$.

Corollary 9.28. *For each level $i \in \{1, \ldots, K\}$ and every node $v \in Z_i$, we have $deg_v(E_i) \leq n^{1/K} \cdot d_i$.*

Proof. Consider two possible cases, depending on the value of i.

- $i = K$.
  In this case, we have $d_i = d_K = n^{1-1/K}$ (see Invariant 9.9), and hence, we get:

$$\deg_v(E_i) \leq |V| = n = n^{1/K} \cdot d_i.$$

- $i \in \{1, \ldots, K-1\}$.
  In this case, applying Lemmas 9.26, 9.27 and Definition 9.8, we get:

$$\deg_v(E_i) \leq \deg_v(H_{i+1}) \leq d_{i+1} = n^{1/K} \cdot d_i.$$

□

9.7 Feasibility of the solution

We devote this section to the proof of Theorem 9.4, which states that the fractional assignment $(w + w_1^* + w^r)$ forms a fractional matching in the graph $G = (V, E)$. Specifically, we will show that $W_v(w + w_1^* + w^r) \leq 1$ for all nodes $v \in V$. The high level idea behind the proof can be explained as follows.

Consider any node $v \in V$ at level $i \in \{1, \ldots, K\}$. We need to show that $W_v(w + w_1^* + w^r) \leq 1$. We start by noting that the quantity $W_v(w + w_1^* + w^r)$ is equal to the total weight received by the node v from all the structures for levels $j \in \{i, \ldots, K\}$. This holds since by Corollary 9.1, we have $v \notin Z_j$ for all $j \in \{1, \ldots, i-1\}$, and hence the node v receives *zero* weight from all the structures for levels $j < i$.

Consider the scenario where we have fixed the structures for all the levels $j \in \{i+1, \ldots, K\}$, and we are about to derive the structures for level i. We first want to upper bound the total weight received by the node v from the residual fractional assignments in levels $j > i$. This is given by $\sum_{j=i+1}^{K} W_v(w_j^r)$. By Invariant 9.14, each w_j^r is a fractional b-matching with respect to the capacities $\{b_j^r(x)\}$. Hence, we infer that $W_v(w_j^r) \leq b_j^r(v)$ for all $j \in \{i+1, \ldots, K\}$. Thus, we get: $\sum_{j>i} W_v(w_j^r) \leq \sum_{j>i} b_j^r(v)$. So it suffices to upper bound the quantity $\sum_{j=i+1}^{K} b_j^r(v)$, which is done in Lemma 9.29. Next, we note that the total weight received by v from the fractional assignments $\{w_j\}$ at levels $j > i$ is $\sum_{j>i} W_v(w_j)$. Thus, we get:

$$\text{Total weight received by } v \text{ from all the structures for levels } j > i \text{ is at most } \sum_{j>i} b_j^r(v) + \sum_{j>i} W_v(w_j). \quad (211)$$

Now, we focus on the weights received by v from the structures at levels i. Here, we need to consider two possible cases, depending on the value of $\ell(v) = i$.

- *Case 1.* $\ell(v) = i \in \{2, \ldots, K\}$.
  In this case, the total weight received by v from the structures at levels i is given by $W_v(w_i) + W_v(w_i^r)$. Since w_i is a fractional b-matching with respect to the capacities $\{b_i(x)\}$, we have $W_v(w_i) \leq b_i(v)$.

Since w_i^r is a fractional b-matching with respect to the capacities $\{b_i^r(x)\}$, we have $W_v(w_i^r) \leq b_i^r(v)$. Thus, we get: $W_v(w_i) + W_v(w_i^r) \leq b_i(v) + b_i^r(v)$. Now, suppose we can prove that:

$$(b_i(v) + b_i^r(v)) + \sum_{j>i} b_j^r(v) + \sum_{j>i} W_v(w_j) \leq 1. \tag{212}$$

Then equations 211 and 212 will together have the following implication: the total weight received by v from all the structures for levels $j \geq i$ is at most one. Since v receives *zero* weight from all the structures for levels $j < i$, it will follow that $W_v(w + w_1^* + w^r)$. Thus, Theorem 9.4 follows from equation 212, which is shown in Lemma 9.30.

- *Case 2.* $\ell(v) = i = 1$.
  In this case, the total weight received by v from the structures at level $i = 1$ is given by $W_v(w_1^*)$. Since w_1^* is a fractional b-matching with respect to the capacities $\{b_1^*(x)\}$, we have $W_v(w_1^*) \leq b_1^*(v)$. Now, suppose we can prove that:

$$b_1^*(v) + \sum_{j=2}^{K} b_1^r(v) + \sum_{j=2}^{K} W_v(w_j) \leq 1. \tag{213}$$

  Then equations 211 and 213 will together have the following implication: the total weight received by v from all the structures for levels $j \in \{1, \ldots, K\}$ is at most one. Thus, Theorem 9.4 follows from equation 213, which is shown in Lemma 9.31.

The rest of this section is organized as follows.

- In Section 9.7.1 we prove Lemma 9.29.
- In Section 9.7.2 we prove Lemma 9.30.
- In Section 9.7.3 we prove Lemma 9.31.
- Finally, in Section 9.7.4 we apply Lemmas 9.29, 9.30 and 9.31 to prove Theorem 9.4.

Lemma 9.29. *Consider any node $v \in V$ at level $\ell(v) = i \in \{1, \ldots, K\}$. We have:*

$$\sum_{j=i+1}^{K} b_j^r(v) \leq \begin{cases} 0 & \text{if } s_i(v) = \infty; \\ 10^{K - s_i(v) + 1} \cdot \delta & \text{else if } s_i(v) \in \{i+1, \ldots, K\}. \end{cases}$$

Lemma 9.30. *Consider any node $v \in V$ at level $\ell(v) = i \in \{2, \ldots, K\}$. We have:*

$$b_i(v) + b_i^r(v) \leq 1 - \sum_{j=i+1}^{K} b_j^r(v) - \sum_{j=i+1}^{K} W_v(w_j).$$

Lemma 9.31. *Consider any node $v \in V$ at level $\ell(v) = 1$. We have:*

$$b_1^*(v) \leq 1 - \sum_{j=2}^{K} b_j^r(v) - \sum_{j=2}^{K} W_v(w_j).$$

9.7.1 Proof of Lemma 9.29

We consider two possible cases, depending on equation 172.

- *Case 1.* $s_i(v) = \infty$.
  Since $\ell(v) = i$, equation 170 and Definitions 9.6, 9.19 imply that $v \in Z_{j-1} = Z_j \setminus T_j$ for all $j \in \{i+1, \ldots, K\}$. Thus, we infer that:

$$v \notin T_j \text{ for all } j \in \{i+1, \ldots, K\}. \tag{214}$$

Since $s_i(v) = \infty$, by equation 172 we have $v \notin S_j$ for all $j \in \{i+1, \ldots, K\}$. This observation, along with equation 214, implies that:

$$v \notin S_j \cup T_j \text{ for all } j \in \{i+1, \ldots, K\}. \tag{215}$$

Equation 215 and Definition 9.13 imply that:

$$b_j^r(v) = 0 \text{ for all } j \in \{i+1, \ldots, K\}. \tag{216}$$

The lemma follows from summing equation 216 over all $j \in \{i+1, \ldots, K\}$.

- *Case 2.* $s_i(v) \in \{i+1, \ldots, K\}$.
  Let $s_i(v) = k$ for some $k \in \{i+1, \ldots, K\}$. Note that k is the minimum index in $\{i+1, \ldots, K\}$ for which $v \in S_k$. Thus, we get:
  $$v \notin S_j \text{ for all } j \in \{i+1, \ldots, k-1\}. \tag{217}$$
  Since $\ell(v) = i$, equation 170 and Definitions 9.6, 9.19 imply that $v \in Z_{j-1} = Z_j \setminus T_j$ for all $j \in \{i+1, \ldots, K\}$. Thus, similar to Case 1, we infer that:
  $$v \notin T_j \text{ for all } j \in \{i+1, \ldots, K\}. \tag{218}$$
  Equations 217 and 218 imply that:
  $$v \notin S_j \cup T_j \text{ for all } j \in \{i+1, \ldots, k-1\}.$$
  Using the above inequality in conjunction with Definition 9.13, we conclude that:
  $$b_j^r(v) = 0 \text{ for all } j \in \{i+1, \ldots, k-1\}. \tag{219}$$
  Now, we focus on the interval $[k, K]$. From equation 218, we get $v \notin T_j$ for all $j \in \{k, \ldots, K\}$. Thus, Definition 9.13 implies that $b_j^r(v) \in \{0, 8 \cdot 10^{K-j} \cdot \delta\}$ for all $j \in \{k, \ldots, K\}$. Thus, we get:
  $$b_j^r(v) \leq 8 \cdot 10^{K-j} \cdot \delta \text{ for all } j \in \{k, \ldots, K\}. \tag{220}$$
  Adding equations 219 and 220 we get:
  $$\begin{aligned} \sum_{j=i+1}^{K} b_j^r(v) &= \sum_{j=i+1}^{k-1} b_j^r(v) + \sum_{j=k}^{K} b_j^r(v) \\ &\leq 0 + \sum_{j=k}^{K} 8 \cdot 10^{K-j} \cdot \delta \\ &\leq 10^{K-k+1} \cdot \delta \\ &= 10^{K-s_i(v)+1} \end{aligned} \tag{221}$$
  This concludes the proof of the lemma in this case.

9.7.2 Proof of Lemma 9.30

Since $\ell(v) = i \geq 2$, Observation 9.25 implies that $v \in T_i$. We now consider two possible cases, depending on the value of $s_i(v)$.

- *Case 1.* $s_i(v) = \infty$.
  In this case, since $v \in T_i$, Definition 9.13 states that:

$$b_i^r(v) = \delta \tag{222}$$

  Further, Definition 9.7 states that:

$$b_i(v) = 1 - \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j) \tag{223}$$

  Invariant 9.10 states that:

$$\mathscr{W}_v(w_j) \geq W_v(w_j) \text{ for all } j \in \{i, \ldots, K\} \tag{224}$$

  From equations 223 and 224, we get:

$$b_i(v) \leq 1 - \delta - \sum_{j=i+1}^{K} W_v(w_j) \tag{225}$$

  Finally, Lemma 9.29 states that:

$$\sum_{j=i+1}^{K} b_j^r(v) \leq 0 \tag{226}$$

  From equations 222, 225 and 226, we get:

$$b_i(v) + b_i^r(v) \leq 1 - \sum_{j=i+1}^{K} b_j^r(v) - \sum_{j=i+1}^{K} W_v(w_j).$$

  This concludes the proof of the lemma in this case.
- *Case 2.* $s_i(v) \in \{i+1, \ldots, K\}$.
  In this case, since $v \in T_i$, Definition 9.13 states that:

$$b_i^r(v) = 4 \cdot 10^{(K - s_i(v) + 1)} \cdot \delta \tag{227}$$

  Further, Definition 9.7 states that:

$$b_i(v) = 1 - 5 \cdot 10^{(K - s_i(v) + 1)} \cdot \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j) \tag{228}$$

  Invariant 9.10 states that:

$$\mathscr{W}_v(w_j) \geq W_v(w_j) \text{ for all } j \in \{i, \ldots, K\}. \tag{229}$$

  From equations 228 and 229, we get:

$$b_i(v) \leq 1 - 5 \cdot 10^{(K - s_i(v) + 1)} \cdot \delta - \sum_{j=i+1}^{K} W_v(w_j) \tag{230}$$

  Finally, since $s_i(v) \in \{i+1, \ldots, K\}$, Lemma 9.29 implies that:

$$\sum_{j=i+1}^{K} b_j^r(v) \leq 10^{(K - s_i(v) + 1)} \cdot \delta \tag{231}$$

  From equations 227, 230 and 231, we get:

$$b_i(v) + b_i^r(v) \leq 1 - \sum_{j=i+1}^{K} b_j^r(v) - \sum_{j=i+1}^{K} W_v(w_j).$$

  This concludes the proof of the lemma in this case.

9.7.3 Proof of Lemma 9.31

We consider two possible cases, depending on the value of $s_1(v)$.

- *Case 1.* $s_1(v) = \infty$.
  In this case, Lemma 9.29 states that:
  $$\sum_{j=2}^{K} b_j^r(v) \leq 0 \tag{232}$$
  Definition 9.20 states that:
  $$b_1^*(v) = 1 - \sum_{j=2}^{K} \mathscr{W}_v(w_j) \tag{233}$$
  Invariant 9.10 states that:
  $$\mathscr{W}_v(w_j) \geq W_v(w_j) \text{ for all } j \in \{2,\ldots,K\}. \tag{234}$$
  From equations 233 and 234, we get:
  $$b_1^*(v) \leq 1 - \sum_{j=2}^{K} W_v(w_j) \tag{235}$$
  From equations 232 and 235, we get:
  $$b_1^*(v) \leq 1 - \sum_{j=2}^{K} b_j^r(v) - \sum_{j=2}^{K} W_v(w_j).$$
  This concludes the proof of the lemma in this case.
- *Case 2.* $s_1(v) \in \{2,\ldots,K\}$.
  In this case, Lemma 9.29 implies that
  $$\sum_{j=2}^{K} b_j^r(v) \leq 10^{(K-s_1(v)+1)} \cdot \delta \tag{236}$$
  Definition 9.20 states that:
  $$b_1^*(v) = 1 - 10^{(K-s_1(v)+1)} \cdot \delta - \sum_{j=2}^{K} \mathscr{W}_v(w_j) \tag{237}$$
  Invariant 9.10 states that:
  $$\mathscr{W}_v(w_j) \geq W_v(w_j) \text{ for all } j \in \{2,\ldots,K\}. \tag{238}$$
  From equations 237 and 238, we get:
  $$b_1^*(v) \leq 1 - 10^{(K-s_1(v)+1)} \cdot \delta - \sum_{j=2}^{K} W_v(w_j) \tag{239}$$
  From equations 236 and 239, we get:
  $$b_1^*(v) \leq 1 - \sum_{j=2}^{K} b_j^r(v) - \sum_{j=2}^{K} W_v(w_j).$$
  This concludes the proof of the lemma in this case.

9.7.4 Proof of Theorem 9.4

Fix any node $v \in V$. We will show that $W_v(w + w_1^* + w^r) \leq 1$. This will imply that $(w + w_1^* + w^r)$ is a fractional matching in the graph $G = (V, E)$. We consider two possible cases, depending on the level of v.

- $\ell(v) = i \in \{2, \ldots, K\}$.
  In this case, applying Lemma 9.30, we get:

$$b_i(v) + \sum_{j=i+1}^{K} W_v(w_j) + \sum_{j=i}^{K} b_j^r(v) \leq 1 \tag{240}$$

  Invariant 9.9 states that w_i is a fractional b-matching with respect to the node-capacities $\{b_i(x)\}$. Hence, we get:

$$W_v(w_i) \leq b_i(v) \tag{241}$$

  Invariant 9.14 states that for all $j \geq i$, w_j^r is a fractional b-matching with respect to the node-capacities $\{b_j^r(x)\}$. Hence, we get:

$$W_v(w_j^r) \leq b_j^r(v) \text{ for all } j \in \{i, \ldots, K\}. \tag{242}$$

  From equations 240, 241 and 242, we get:

$$\sum_{j=i}^{K} W_v(w_j) + \sum_{j=i}^{K} W_v(w_j^r) \leq 1 \tag{243}$$

  Since $\ell(v) = i \in \{2, \ldots, K\}$, we have $v \notin Z_j$ for all $j \in \{1, \ldots, i-1\}$. Thus, we conclude that:

$$W_v(w_j) = W_v(w_j^r) = 0 \text{ for all } j \in \{2, \ldots, i-1\}; \text{ and } W_v(w_1^*) = 0. \tag{244}$$

  From equations 243 and 244, we get:

$$\begin{aligned} W_v(w + w_1^* + w_r) &= W_v(w_1^*) + W_v(w) + W_v(w^r) \\ &= W_v(w_1^*) + \sum_{j=2}^{i-1} \left(W_v(w_j) + W_v(w_j^r)\right) + \sum_{j=i}^{K} \left(W_v(w_j) + W_v(w_j^r)\right) \\ &\leq 0 + 0 + 1 \\ &= 1 \end{aligned}$$

  This concludes the proof of the lemma in this case.

- $\ell(v) = 1$.
  In this case, applying Lemma 9.31, we get:

$$b_1^*(v) + \sum_{j=2}^{K} W_v(w_j) + \sum_{j=2}^{K} b_j^r(v) \leq 1 \tag{245}$$

  Invariant 9.21 states that w_1^* is a fractional b-matching with respect to the capacities $\{b_1^*(x)\}$. Hence, we get:

$$W_v(w_1^*) \leq b_1^*(v) \tag{246}$$

  Invariant 9.14 states that for all $j \in \{2, \ldots, K\}$, w_j^r is a fractional b-matching with respect to the capacities $\{b_j^r(x)\}$. Hence, we get:

$$W_v(w_j^r) \leq b_j^r(v) \text{ for all } j \in \{2, \ldots, K\}. \tag{247}$$

From equations 245, 246 and 247, we get:

$$\begin{aligned} W_v(w+w_1^*+w^r) &= W_v(w_1^*)+W_v(w)+W_v(w^r) \\ &= W_v(w_1^*)+\sum_{j=2}^{K} W_v(w_j)+\sum_{j=2}^{K} W_v(w^r) \\ &\leq b_1^*(v)+\sum_{j=2}^{K} W_v(w_j)+\sum_{j=2}^{K} b_j^r(v) \\ &\leq 1. \end{aligned}$$

This concludes the proof of the lemma in this case.

9.8 Approximation Guarantee of the Solution

We want to show that the size of the fractional assignment $(w+w_1^*+w^r)$ is *strictly* within a factor of two of the maximum cardinality matching in the input graph $G=(V,E)$. So we devote this section to the proof of Theorem 9.5. We start by recalling some notations that will be used throughout the rest of this section.

- Given any subset of edges $E' \subseteq E$, we let $V(E') = \{u \in V : \deg_{E'}(v) > 0\}$ be the set of endpoints of the edges in E'. Thus, for a matching $M \subseteq E$, the set of matched nodes is given by $V(M)$.
- Consider any level $i \in \{2,\ldots,K\}$, and recall that Invariant 9.10 introduces the concept of a "discretized" node-weight $\mathscr{W}_v(w_i)$ for all $v \in Z_i$. We "extend" this definition as follows. If a node v belongs to a level $\ell(v) = i \in \{2,\ldots,K\}$, then $v \notin Z_j$ for all $j \in \{1,\ldots,i-1\}$. In this case, we set $\mathscr{W}_v(w_j) = W_v(w_j) = 0$ for all $j \in \{1,\ldots,i-1\}$. Further, for every node $u \in V$, we define $\mathscr{W}_u(w) = \sum_{j=2}^{K} \mathscr{W}_u(w_j)$.

Instead of directly trying to bound the size of the fractional assignment $(w+w_1^*+w^r)$, we first prove the following theorem. Fix any matching $M^* \subseteq E$ in the input graph G. The theorem below upper bounds the size of this matching in terms of the total discretized node-weight received by the matched nodes, plus the total weight received by all the nodes from w_1^* and w^r. The proof of Theorem 9.32 appears in Section 9.8.1.

Theorem 9.32. *Consider any matching $M^* \subseteq E$ in the input graph $G=(V,E)$. Then we have:*

$$\sum_{v \in V(M^*)} \mathscr{W}_v(w) + \sum_{v \in V} (W_v(w_1^*)+W_v(w^r)) \geq (1+\varepsilon)^{-1} \cdot (1+\delta/3) \cdot |M^*|.$$

To continue with the proof of Theorem 9.5, we need the observation that the discretized weight of a node is very close to its normal weight. This intuition is quantified in the following claim.

Claim 9.33. *For every node $v \in V$, we have $W_v(w) \geq \mathscr{W}_v(w) - 2K\varepsilon$.*

Proof. Invariant 9.10 implies that:

$$W_v(w_j) \geq \mathscr{W}_v(w_j) - 2\varepsilon \text{ for all } j \in \{2,\ldots,K\}.$$

Summing the above inequality over all levels $j \in \{2,\ldots,K\}$, we get:

$$\begin{aligned} W_v(w) &= \sum_{j=2}^{K} W_v(w_j) \\ &\geq \sum_{j=2}^{K} (\mathscr{W}_v(w_j) - 2\varepsilon) \\ &= \sum_{j=2}^{K} \mathscr{W}_v(w_j) - 2\varepsilon(K-1) \\ &= \mathscr{W}_v(w) - 2\varepsilon(K-1) \geq \mathscr{W}_v(w) - 2\varepsilon K. \end{aligned}$$

□

We can now lower bound the total weight received by all the nodes under $(w+w_1^*+w^r)$ as follows.

Claim 9.34. *Consider any matching $M^* \subseteq E$ in the input graph $G=(V,E)$. We have:*

$$\sum_{v\in V} W_v(w+w_1^*+w^r) \geq \left(\frac{(1+\delta/3)}{(1+\varepsilon)} - 4K\varepsilon\right)\cdot|M^*|.$$

Proof. We infer that:

$$\begin{aligned}
\sum_{v\in V} W_v(w+w_1^*+w^r) &= \sum_{v\in V} W_v(w) + \sum_{v\in V} W_v(w_1^*) + \sum_{v\in V} W_v(w^r) \\
&\geq \sum_{v\in V(M^*)} W_v(w) + \sum_{v\in V} W_v(w_1^*) + \sum_{v\in V} W_v(w^r) && (248)\\
&\geq \sum_{v\in V(M^*)} (\mathscr{W}_v(w) - 2\varepsilon K) + \sum_{v\in V} W_v(w_1^*) + \sum_{v\in V} W_v(w^r) && (249)\\
&= \sum_{v\in V(M^*)} \mathscr{W}_v(w) - 2\cdot|M^*|\cdot(2\varepsilon K) + \sum_{v\in V} W_v(w_1^*) + \sum_{v\in V} W_v(w^r) && (250)\\
&= \left(\sum_{v\in V(M^*)} \mathscr{W}_v(w) + \sum_{v\in V} W_v(w_1^*) + \sum_{v\in V} W_v(w^r)\right) - (4\varepsilon K)\cdot|M^*| \\
&\geq \frac{(1+\delta/3)}{(1+\varepsilon)}\cdot|M^*| - (4\varepsilon K)\cdot|M^*| && (251)\\
&= \left(\frac{(1+\delta/3)}{(1+\varepsilon)} - 4\varepsilon K\right)\cdot|M^*| && (252)
\end{aligned}$$

Equation 248 holds since $V(M^*)\subseteq V$. Equation 249 follows from Claim 9.33. Equation 250 holds since $|V(M^*)| = 2\cdot|M^*|$ as no two edges in M^* share a common endpoint. Equation 251 follows from Theorem 9.32. Finally, the claim follows from equation 252. □

We now use Claim 9.34 to lower bound the size of the fractional assignment $(w+w_1^*+w^r)$ by relating the sum of the node-weights with the sum of the edge-weights. Specifically, we note that each edge $e\in E$ contributes $2\cdot(w(e)+w_1^*(e)+w^r(e))$ towards the sum $\sum_{v\in V} W_v(w+w_1^*+w^r)$. Hence, we infer that the sum of the edge-weights under $(w+w_1^*+w^r)$ is exactly $(1/2)\cdot\sum_{v\in V} W_v(w+w_1^*+w^r)$.

$$\sum_{e\in E}(w(e)+w_1^*(e)+w^r(e)) = (1/2)\cdot\sum_{v\in V} W_v(w+w_1^*+w^r) \tag{253}$$

Accordingly, Claim 9.34 and equation 253 imply that the size of the fractional assignment $(w+w_1^*+w^r)$ is at least f times the size of any matching $M^*\subseteq E$ in $G=(V,E)$, where:

$$f = (1/2)\cdot\left(\frac{(1+\delta/3)}{(1+\varepsilon)} - 4\varepsilon K\right).$$

Setting M^* to be the maximum cardinality matching in G, we derive the proof of Theorem 9.5. To summarize, in order to prove Theorem 9.5, it suffices to prove Theorem 9.32. This is done in Section 9.8.1.

9.8.1 Proof of Theorem 9.32

Define the "level" of an edge to be the maximum level among its endpoints, i.e., we have:

$$\ell(u,v) = \max(\ell(u),\ell(v)) \text{ for every edge } (u,v)\in E. \tag{254}$$

Now, for every $i\in\{1,\ldots,K\}$, let $M_i^* = \{(u,v)\in M^* : \ell(u,v)=i\}$ denote the subset of those edges in M^* that are at level i. It is easy to check that the subsets $M_1^*,\ldots,M_K^*$ partition the edge-set M^*.

By Observation 9.25, if a node $v \in V$ is at level $\ell(v) = i \in \{2,\ldots,K\}$, then we also have $v \in T_i$. Consider any edge $(x,y) \in M_i^*$, where $i \geq 2$. Without any loss of generality, suppose that $\ell(x) = i$ and hence $x \in T_i$. Consider the other endpoint y of this matched edge (x,y). By definition, we have $\ell(y) \leq i$, which means that $y \in Z_i$. Now, Invariant 9.12 states that the set Z_i is partitioned into subsets T_i, B_i and S_i. Thus, there are two mutually exclusive cases to consider: (1) either $y \in T_i \cup B_i$ or (2) $y \in S_i$. Accordingly, we partition the edge-set M_i^* into two subsets – M_i' and M_i'' – as defined below.

1. $M_i' = \{(u,v) \in M_i^* : u \in T_i \text{ and } v \in T_i \cup B_i\}$. This consists of the set of matched edges in M_i^* with one endpoint in T_i and the other endpoint in $T_i \cup B_i$.
2. $M_i'' = \{(u,v) \in M_i^* : u \in T_i \text{ and } v \in S_i\}$. This consists of the set of matched edges in M_i^* with one endpoint in T_i and the other endpoint in S_i.

To summarize, the set of edges in M^* is partitioned into the following subsets:

$$M_1^*, \{M_2', M_2''\}, \{M_3', M_3''\}, \ldots, \{M_K', M_K''\}.$$

We now define the matchings M' and M'' as follows.

$$M' = \bigcup_{i=2}^{K} M_i' \text{ and } M'' = \bigcup_{i=2}^{K} M_i'' \tag{255}$$

Hence, the set of matched edges M^* is partitioned by the subsets M_1^*, M' and M''. We will now consider the matchings M_1^*, M' and M'' one after the other, and upper bound their sizes in terms of the node-weights.

Lemma 9.35 helps us bound the size of the matching M_1^*. Note that each edge $(u,v) \in M_1^*$ has $\ell(u,v) = \max(\ell(u),\ell(v)) = 1$, and hence we must have $\ell(u) = \ell(v) = 1$. In other words, every edge in M_1^* has both of its endpoints in level one. Since E_1 is the set of edges connecting the nodes at level one (see Definition 9.19), we infer that $M_1^* \subseteq E_1$. We will later apply Lemma 9.35 by setting $M = M_1^*$. The proof of Lemma 9.35 appears in Section 9.8.2.

Lemma 9.35. *Consider any matching $M \subseteq E_1$ in the subgraph $G_1 = (Z_1, E_1)$. Then we have:*

$$\sum_{v \in V(M)} \mathscr{W}_v(w) + \sum_{v \in Z_1} W_v(w_1^*) \geq (1+\varepsilon)^{-1} \cdot (1+\delta) \cdot |M|.$$

Lemma 9.36 helps us bound the size of the matching M_i', for all $i \in \{2,\ldots,K\}$. Note that by definition every edge $(u,v) \in M_i'$ has one endpoint $u \in T_i$ and the other endpoint $v \in T_i \cup B_i$. Thus, we will later apply Lemma 9.36 by setting $M = M_i'$. The proof of Lemma 9.36 appears in Section 9.8.3.

Lemma 9.36. *Consider any level $i \in \{2,\ldots,K\}$. Let $M \subseteq E_i$ be a matching in the subgraph $G_i = (Z_i, E_i)$ such that every edge $(u,v) \in M$ has one endpoint $u \in T_i$ and the other endpoint $v \in T_i \cup B_i$. Then we have:*

$$\sum_{u \in V(M)} \mathscr{W}_u(w) \geq (1+3\delta) \cdot |M|.$$

Lemma 9.37 helps us bound the size of the matching M_i'', for all $i \in \{2,\ldots,K\}$. Note that by definition every edge $(u,v) \in M_i''$ has one endpoint $u \in T_i$ and the other endpoint $v \in S_i$. Thus, we can apply Lemma 9.37 by setting $M = M_i''$. The proof of Lemma 9.37 appears in Section 9.8.4.

Lemma 9.37. *Consider any level $i \in \{2,\ldots,K\}$. Let $M \subseteq E_i$ be a matching in the subgraph $G_i = (Z_i, E_i)$ such that every edge $(u,v) \in M$ has one endpoint $u \in T_i$ and the other endpoint $v \in S_i$. We have:*

$$\sum_{u \in V(M)} \mathscr{W}_u(w) + \sum_{v \in Z_i} W_v(w_i^r) \geq (1+\delta/3) \cdot |M|.$$

To proceed with the proof of Theorem 9.32, we first apply Lemma 9.35 to get the following simple claim. This claim upper bounds the size of M_1^* in terms of the total discretized weight received by its endpoints, plus the total weight received from w_1^* by the nodes in V.

Claim 9.38. *We have:*

$$\sum_{v \in V(M_1^*)} \mathscr{W}_v(w) + \sum_{v \in V} W_v(w_1^*) \geq (1+\varepsilon)^{-1} \cdot (1+\delta) \cdot |M_1^*|.$$

Proof. Since M_1^* is a matching in $G_1 = (Z_1, E_1)$, from Lemma 9.35, we get:

$$\sum_{v \in V(M_1^*)} \mathscr{W}_v(w) + \sum_{v \in Z_1} W_v(w_1^*) \geq (1+\varepsilon)^{-1} \cdot (1+\delta) \cdot |M_1^*|.$$

Since $Z_1 \subseteq V$, the claim follows from the above inequality. □

Recall that the edge-set M' is partitioned into subsets $M'_2, \ldots, M'_K$. Thus, we can upper bound $|M'|$ by applying Lemma 9.36 with $M = M'_i$, for $i \in \{2, \ldots, K\}$, and summing over the resulting inequalities. This is done in the claim below. This claim upper bounds the size of M' in terms of the total discretized weight received by its endpoints.

Claim 9.39. *We have:*

$$\sum_{v \in V(M')} \mathscr{W}_v(w) \geq (1+3\delta) \cdot |M'|.$$

Proof. For every $i \in \{2, \ldots, K\}$, we can apply Lemma 9.36 on the matching M'_i to get:

$$\sum_{v \in V(M'_i)} \mathscr{W}_v(w) \geq (1+3\delta) \cdot |M'_i| \tag{256}$$

Now, note that the set of edges M' has been partitioned into subsets $M'_2, \ldots, M'_K$. Hence, summing equation 256 over all levels $2 \leq i \leq K$, we get:

$$\begin{aligned} \sum_{v \in V(M')} \mathscr{W}_v(w) &= \sum_{i=2}^{K} \sum_{v \in V(M'_i)} \mathscr{W}_v(w) \\ &\geq \sum_{i=2}^{K} (1+3\delta) \cdot |M'_i| \\ &= (1+3\delta) \cdot |M'| \end{aligned}$$

□

Recall that the edge-set M'' is partitioned into subsets $M''_2, \ldots, M''_K$. Further, we have $w^r = \sum_{j=2}^{K} w^r_j$. Thus, we can apply Lemma 9.37 with $M = M''_i$, for $i \in \{2, \ldots, K\}$, and sum over the resulting inequalities to get the following claim. This claim upper bounds the size of M'' in terms of the total discretized weight received by its endpoints, plus the total weight received from w^r by the nodes in V.

Claim 9.40. *We have:*

$$\sum_{v \in V(M'')} \mathscr{W}_v(w) + \sum_{v \in V} W_v(w^r) \geq (1+\delta/3) \cdot |M''|.$$

Proof. For every $i \in \{2,\ldots,K\}$, since M_i'' is a matching in $G_i = (Z_i, E_i)$ such that each edge $(u,v) \in M_i''$ has one endpoint in T_i and the other endpoint in S_i, we can apply Lemma 9.37 on M_i'' to get:

$$\sum_{v \in V(M_i'')} \mathscr{W}_v(w) + \sum_{v \in Z_i} W_v(w_i^r) \geq (1+\delta/3) \cdot |M_i''|.$$

Since $Z_i \subseteq V$, we have $\sum_{v \in V} W_v(w_i^r) \geq \sum_{v \in Z_i} W_v(w_i^r)$. Hence, the above inequality implies that:

$$\sum_{v \in V(M_i'')} \mathscr{W}_v(w) + \sum_{v \in V} W_v(w_i^r) \geq (1+\delta/3) \cdot |M_i''| \text{ for all } i \in \{2,\ldots,K\}. \tag{257}$$

Since the set of edges M'' is partitioned into subsets $M_2'',\ldots,M_K''$, and since $w^r = \sum_{i=2}^{K} w_i^r$, we get:

$$\begin{aligned}
\sum_{v \in V(M'')} \mathscr{W}_v(w) + \sum_{v \in V} W_v(w^r) &= \sum_{i=2}^{K} \sum_{v \in V(M_i'')} \mathscr{W}_v(w) + \sum_{v \in V} \sum_{i=2}^{K} W_v(w_i^r) \\
&= \sum_{i=2}^{K} \left(\sum_{v \in V(M_i'')} \mathscr{W}_v(w) + \sum_{v \in V} W_v(w_i^r) \right) \\
&\geq \sum_{i=2}^{K} (1+\delta/3) \cdot |M_i''| \qquad (258) \\
&= (1+\delta/3) \cdot |M''|.
\end{aligned}$$

Equation 258 follows from equation 257. This concludes the proof of the claim. □

Since the set of matched edges M^* is partitioned into subsets M_1^*, M' and M'', we can now add the inequalities in Claims 9.38, 9.39 and 9.40 to derive Theorem 9.32. The primary observation is that the sum in the left hand side of Theorem 9.32 upper bounds the sum of the left hand sides of the inequalities stated in these three claims. This holds since no two edges in M^* share a common endpoint, and, accordingly, no amount of node-weight is counted twice if we sum the left hand sides of these claims. To bound the sum of the right hand sides of these claims, we use the fact that $|M^*| = |M_1^*| + |M'| + |M''|$. Specifically, we get:

$$\begin{aligned}
&\sum_{v \in V(M)} \mathscr{W}_v(w) + \sum_{v \in V} (W_v(w_1^*) + W_v(w^r)) \\
= &\sum_{v \in V(M)} \mathscr{W}_v(w) + \sum_{v \in V} W_v(w_1^*) + \sum_{v \in V} W_v(w^r) \\
= &\left(\sum_{v \in V(M_1^*)} \mathscr{W}_v(w) + \sum_{v \in V(M')} \mathscr{W}_v(w) + \sum_{v \in V(M'')} \mathscr{W}_v(w) \right) + \sum_{v \in V} W_v(w_1^*) + \sum_{v \in V} W_v(w^r) \qquad (259) \\
= &\left(\sum_{v \in V(M_1^*)} \mathscr{W}_v(w) + \sum_{v \in V} W_v(w_1^*) \right) + \left(\sum_{v \in V(M'')} \mathscr{W}_v(w) + \sum_{v \in V} W_v(w^r) \right) + \sum_{v \in V(M')} \mathscr{W}_v(w) \\
\geq &\ (1+\varepsilon)^{-1} \cdot (1+\delta) \cdot |M_1^*| + (1+\delta/3) \cdot |M''| + (1+3\delta) \cdot |M'| \qquad (260) \\
\geq &\ (1+\varepsilon)^{-1} \cdot (1+\delta/3) \cdot \left(|M_1^*| + |M''| + |M'| \right) \\
= &\ (1+\varepsilon)^{-1} \cdot (1+\delta/3) \cdot |M^*| \qquad (261)
\end{aligned}$$

Equation 259 holds since the edges in the matching M^* are partitioned into subsets M_1^*, M', M'', and hence, the node-set $V(M^*)$ is also partitioned into subsets $V(M_1^*), V(M'), V(M'')$. Equation 260 follows from Claims 9.38, 9.39 and 9.40. Finally, Theorem 9.32 follows from equation 261.

9.8.2 Proof of Lemma 9.35

We will carefully construct a fractional b-matching w_1' in the graph $G_1 = (Z_1, E_1)$ with respect to the node-capacities $\{b_1^*(u)\}, u \in Z_1$. Then we will show that:

$$\sum_{v \in V(M)} \mathscr{W}_v(w) + \sum_{v \in Z_1} W_v(w_1') \geq (1+\delta) \cdot |M| \tag{262}$$

By Invariant 9.21, we know that w_1^* is a fractional b-matching in the graph G_1 with respect to the node-capacities $\{b_1^*(u)\}, u \in Z_1$. Further, the size of w_1^* is a $(1+\varepsilon)$-approximation to the size of the maximum fractional b-matching with respect to the same node-capacities. Thus, we have:

$$\sum_{v \in Z_1} W_v(w_1^*) \geq (1+\varepsilon)^{-1} \cdot \sum_{v \in Z_1} W_v(w_1') \tag{263}$$

From equations 262 and 263, we can derive that:

$$\begin{aligned} \sum_{v \in V(M)} \mathscr{W}_v(w) + \sum_{v \in Z_1} W_v(w_1^*) &\geq \sum_{v \in V(M)} \mathscr{W}_v(w) + (1+\varepsilon)^{-1} \cdot \sum_{v \in Z_1} W_v(w_1') \\ &\geq (1+\varepsilon)^{-1} \cdot (1+\delta) \cdot |M| \end{aligned}$$

Thus, the lemma follows from equations 262 and 263.

It remains to prove equation 262. Towards this end, we first define the fractional assignment w_1' below.

$$w_1'(u,v) = \begin{cases} \min(b_1^*(u), b_1^*(v)) & \text{for all edges } (u,v) \in M; \\ 0 & \text{for all edges } (u,v) \in E_1 \setminus M. \end{cases} \tag{264}$$

Since $M \subseteq E_1$ and since no two edges in M share a common endpoint, w_1' as defined above forms a fractional b-matching in the graph $G_1 = (Z_1, E_1)$ with respect to the node-capacities $\{b_1^*(u)\}, u \in Z_1$. We will show that:

$$\mathscr{W}_x(w) + W_x(w_1') + \mathscr{W}_y(w) + W_y(w_1') \geq (1+\delta) \text{ for every edge } (x,y) \in M. \tag{265}$$

By equation 264 we have $W_v(w_1') = 0$ for all nodes $v \in Z_1 \setminus V(M)$. Hence, equation 265 implies that:

$$\sum_{v \in V(M)} \mathscr{W}_v(w) + \sum_{v \in Z_1} W_v(w_1') \geq \sum_{(x,y) \in M} \left\{ \mathscr{W}_x(w) + W_x(w_1') + \mathscr{W}_y(w) + W_y(w_1') \right\} \geq (1+\delta) \cdot |M|$$

Hence, equation 262 follows from equation 265. We thus focus on showing that equation 265 holds.

Proof of equation 265.

Fix any edge $(x,y) \in M$, and suppose that $b_1^*(x) \leq b_1^*(y)$. Then equation 264 implies that:

$$W_x(w_1') = W_y(w_1') = w_1'(x,y) = b_1^*(x) \tag{266}$$

We claim that:

$$\mathscr{W}_x(w) + 2 \cdot b_1^*(x) \geq 1 + \delta \tag{267}$$

From equations 266 and 267, we can make the following deductions.

$$\begin{aligned} \mathscr{W}_x(w) + W_x(w_1') + \mathscr{W}_y(w) + W_y(w_1') &\geq \mathscr{W}_x(w) + W_x(w_1') + W_y(w_1') \\ &= \mathscr{W}_x(w) + 2 \cdot b_1^*(x) \\ &\geq 1 + \delta \end{aligned}$$

Thus, in order to prove equation 265, it suffices to prove equation 267. For the rest of this section, we focus on proving equation 267. There are three possible cases to consider, depending on the value of $s_1(x)$.

- *Case 1.* $s_1(x) = \infty$.
  Since $s_1(x) = \infty$, we have $x \notin S_i$ for all $i \in \{2, \ldots, K\}$. In particular, we have $s_2(x) = \infty$. Hence, Definition 9.7 implies that:
  $$b_2(x) = 1 - \delta - \sum_{j=3}^{K} \mathscr{W}_x(w_j) \tag{268}$$
  Invariant 9.10 implies that:
  $$\mathscr{W}_x(w_2) \leq b_2(x) \tag{269}$$
  From equations 268 and 269, we get:
  $$\mathscr{W}_x(w_2) \leq 1 - \delta - \sum_{j=3}^{K} \mathscr{W}_x(w_j) \tag{270}$$
  Rearranging the terms in the above inequality, we get:
  $$\sum_{j=2}^{K} \mathscr{W}_x(w_j) \leq 1 - \delta \tag{271}$$
  Since $s_1(x) = \infty$, Definition 9.20 imply that:
  $$b_1^*(x) = 1 - \sum_{j=2}^{K} \mathscr{W}_x(w_j) \tag{272}$$
  From equations 271 and 272, we get:
  $$b_1^*(x) \geq \delta \tag{273}$$
  From equations 272 and 273, we get:
  $$2 \cdot b_1^*(x) + \sum_{j=2}^{K} \mathscr{W}_x(w_j) \geq (1 + \delta). \tag{274}$$
  Equation 267 follows from equation 274 and the fact that $\mathscr{W}_x(w) = \sum_{j=2}^{K} \mathscr{W}_x(w_j)$.
- *Case 2.* $s_1(x) \in \{3, \ldots, K\}$.
  Since $s_1(x) \in \{3, \ldots, K\}$, we infer that $s_2(x) = s_1(x)$ (see equation 172). Hence, Definition 9.7 implies that:
  $$b_2(x) = 1 - 5 \cdot 10^{(K - s_1(x) + 1)} \cdot \delta - \sum_{j=3}^{K} \mathscr{W}_x(w_j) \tag{275}$$
  Invariant 9.10 states that:
  $$\mathscr{W}_x(w_2) \leq b_2(x) \tag{276}$$
  From equations 275 and 276, we get:
  $$\mathscr{W}_x(w_2) \leq 1 - 5 \cdot 10^{(K - s_1(x) + 1)} \cdot \delta - \sum_{j=3}^{K} \mathscr{W}_x(w_j) \tag{277}$$
  Rearranging the terms in the above inequality, and noting that $\mathscr{W}_x(w) = \sum_{j=2}^{K} \mathscr{W}_x(w_j)$, we get:
  $$\mathscr{W}_x(w) \leq 1 - 5 \cdot 10^{(K - s_1(x) + 1)} \cdot \delta \tag{278}$$

Since $s_1(x) \in \{3, \ldots, K\}$, Definition 9.20 imply that:

$$b_1^*(x) = 1 - 10^{(K-s_1(x)+1)} \cdot \delta - \mathscr{W}_x(w) \tag{279}$$

From equations 278 and 279, we infer that:

$$b_1^*(x) \geq 4 \cdot 10^{(K-s_1(x)+1)} \cdot \delta \tag{280}$$

From equations 279 and 280, we get:

$$\begin{aligned} \mathscr{W}_x(w) + 2 \cdot b_1^*(x) &= (\mathscr{W}_x(w) + b_1^*(x)) + b_1^*(x) \\ &= \left(1 - 10^{(K-s_1(x)+1)} \cdot \delta\right) + b_1^*(x) \\ &\geq \left(1 - 10^{(K-s_1(x)+1)} \cdot \delta\right) + 4 \cdot 10^{(K-s_1(x)+1)} \cdot \delta \\ &= 1 + 3 \cdot 10^{(K-s_1(x)+1)} \cdot \delta \\ &\geq (1+\delta). \end{aligned}$$

Thus, equation 267 holds in this case.

- *Case 3.* $s_1(x) = 2$.
  Since $s_1(x) = 2$, we have $x \in S_2$. Further, since $\mathscr{W}_x(w) = \sum_{j=2}^{K} \mathscr{W}_x(w_j)$, Definition 9.11 gives:

$$\mathscr{W}_x(w) \leq 8 \cdot 10^{(K-2)} \cdot \delta \tag{281}$$

On the other hand, since $\mathscr{W}_x(w) = \sum_{j=2}^{K} \mathscr{W}_x(w_j)$ and $s_1(x) = 2$, Definition 9.20 states that:

$$b_1^*(x) = 1 - 10^{(K-1)} \cdot \delta - \mathscr{W}_x(w) \tag{282}$$

From equations 281 and 282, we get:

$$\begin{aligned} 2 \cdot b_1^*(x) &= 2 - 2 \cdot 10^{(K-1)} \cdot \delta - 2 \cdot \mathscr{W}_x(w) \\ &\geq 2 - 2 \cdot 10^{(K-1)} \cdot \delta - 16 \cdot 10^{(K-2)} \cdot \delta \\ &= 2 - 36 \cdot 10^{(K-2)} \cdot \delta \\ &\geq 1 + \delta \end{aligned} \tag{283}$$

Equation 283 holds due to equation 163, which implies that $\delta < 1/(1 + 36 \cdot 10^{(K-2)})$. From equation 283, we get:

$$\mathscr{W}_x(w) + 2 \cdot b_1^*(x) \geq 2 \cdot b_1^*(x) \geq (1+\delta).$$

Thus, equation 267 holds in this case.

9.8.3 Proof of Lemma 9.36

Consider any edge $(u,v) \in M$ with $v \in T_i$ and $u \in T_i \cup B_i$. Since $v \in T_i$, Definition 9.11 implies that:

$$\mathscr{W}_v(w_i) = b_i(v) \tag{284}$$

Next, since $i \in \{2, \ldots, K\}$, and either $s_i(v) = \infty$ or $i+1 \leq s_i(v) \leq K$, Definition 9.7 implies that:

$$b_i(v) \geq 1 - 5 \cdot 10^{(K-i)} \cdot \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j) \tag{285}$$

From equations 284 and 285, we infer that:

$$\mathscr{W}_v(w_i) \geq 1 - 5 \cdot 10^{(K-i)} \cdot \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j).$$

Rearranging the terms in the above inequality, we get:

$$\mathscr{W}_v(w) \geq \sum_{j=i}^{K} \mathscr{W}_v(w_j) \geq 1 - 5 \cdot 10^{(K-i)} \cdot \delta \tag{286}$$

Since $u \in T_i \cup B_i$, and since the node-set Z_i is partitioned by the sets T_i, B_i, S_i (see Invariant 9.12), we infer that $u \notin S_i$. Hence, applying Definition 9.11, we get:

$$\mathscr{W}_u(w) \geq \sum_{j=i}^{K} \mathscr{W}_u(w_j) > 8 \cdot 10^{(K-i)} \cdot \delta \tag{287}$$

Adding equations 286 and 287, we get:

$$\mathscr{W}_u(w) + \mathscr{W}_v(w) \geq 1 + 3 \cdot 10^{(K-i)} \cdot \delta \geq 1 + 3\delta \tag{288}$$

Summing equation 288 over all the edges in the matching M, we get:

$$\sum_{x \in V(M)} \mathscr{W}_x(w) = \sum_{(x,y) \in M} (\mathscr{W}_x(w) + \mathscr{W}_y(w)) \geq (1 + 3\delta) \cdot |M|.$$

This concludes the proof of the lemma.

9.8.4 Proof of Lemma 9.37

Set $\lambda = 4 \cdot 10^{K-i} \cdot \delta$. For every node $v \in T_i$, either $s_i(v) = \infty$ or $s_i(v) \in \{i+1, \ldots, K\}$. Thus, Definition 9.13 states that for every node $v \in T_i$, either $b_i^r(v) = \delta \leq \lambda$ or $b_i^r(v) = 4 \cdot 10^{K - s_i(v) + 1} \cdot \delta \leq \lambda$. We infer that $b_i^r(v) \in [0, \lambda]$ for every node $v \in T_i$. In contrast, for every node $v \in S_i$, Definition 9.13 states that $b_i^r(v) = 2\lambda$. By Invariant 9.12, the node-sets $T_i, S_i \subseteq Z_i$ are mutually disjoint. Finally, by Invariant 9.14, the fractional assignment w_i^r is a maximal fractional b-matching in the residual graph $G_i^r = (T_i \cup S_i, E_i^r)$ with respect to the capacities $\{b_i^r(v)\}, v \in T_i \cup S_i$. The residual graph G_i^r is bipartite since $S_i \cap T_i = \emptyset$ and since every edge $e \in E_i^r$ has one endpoint in T_i and the other endpoint in S_i. Further, M is a matching in the graph G_i^r since E_i^r contains all the edges in E_i with one tight and one small endpoints. Thus, Theorem 8.3 implies that:

$$\sum_{v \in T_i \cup S_i} W_v(w_i^r) \geq (4/3) \cdot \sum_{v \in T_i \cap V(M)} b_i^r(v) \tag{289}$$

We can now infer that:

$$\begin{aligned}
\sum_{v \in V(M)} \mathscr{W}_v(w) + \sum_{v \in Z_i} W_v(w_i^r) &\geq \sum_{v \in V(M)} \mathscr{W}_v(w) + \sum_{v \in T_i \cup S_i} W_v(w_i^r) && (290)\\
&\geq \sum_{v \in V(M)} \mathscr{W}_v(w) + (4/3) \cdot \sum_{v \in T_i \cap V(M)} b_i^r(v) && (291)\\
&\geq \sum_{v \in T_i \cap V(M)} \mathscr{W}_v(w) + (4/3) \cdot \sum_{v \in T_i \cap V(M)} b_i^r(v)\\
&= \sum_{v \in T_i \cap V(M)} (\mathscr{W}_v(w) + (4/3) \cdot b_i^r(v))\\
&\geq \sum_{v \in T_i \cap V(M)} (1 + \delta/3) && (292)\\
&= (1 + \delta/3) \cdot |M| && (293)
\end{aligned}$$

Equation 290 holds since $Z_i \supseteq T_i \cup S_i$ (see Invariant 9.12). Equation 291 follows from equation 289. Equation 292 follows from Claim 9.41 stated below. Equation 293 holds since every edge in M has exactly one endpoint in T_i and no two edges in M share a common endpoint. Thus, we have $|T_i \cap V(M)| = |M|$. This leads to the proof of the lemma.

Thus, in order to prove Lemma 9.37, it suffices to prove Claim 9.41. This is done below.

Claim 9.41. *For every node $v \in T_i$, we have: $\mathscr{W}_v(w) + (4/3) \cdot b_i^r(v) \geq 1 + \delta/3$.*

Proof. We consider two possible cases, depending on the value of $s_i(v)$.

- *Case 1.* $s_i(v) = \infty$.
  In this case, Definition 9.13 states that:
  $$b_i^r(v) = \delta \tag{294}$$
  On the other hand, since $v \in T_i$, Definition 9.11 states that:
  $$\mathscr{W}_v(w_i) = b_i(v) \tag{295}$$
  Finally, since $v \in Z_i$ and $s_i(v) = \infty$, Definition 9.7 states that:
  $$b_i(v) = 1 - \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j) \tag{296}$$
  From equations 295 and 296 we get:
  $$\mathscr{W}_v(w_i) = 1 - \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j).$$
  Rearranging the terms in the above inequality, we get:
  $$\mathscr{W}_v(w) \geq \sum_{j=i}^{K} \mathscr{W}_v(w_j) = 1 - \delta \tag{297}$$
  From equations 294 and 297, we infer that:
  $$\mathscr{W}_v(w) + (4/3) \cdot b_i^r(v) \geq (1 + \delta/3).$$
  This concludes the proof of the claim in this case.
- *Case 2.* $s_i(v) = k$ for some $k \in \{i+1, \ldots, K\}$.
  In this case, Definition 9.13 states that:
  $$b_i^r(v) = 4 \cdot 10^{(K-k+1)} \cdot \delta \tag{298}$$
  On the other hand, since $v \in T_i$, Definition 9.11 states that:
  $$\mathscr{W}_v(w_i) = b_i(v) \tag{299}$$
  Finally, since $v \in Z_i$ and $s_i(v) = k \in \{i+1, \ldots, K\}$, Definition 9.7 states that:
  $$b_i(v) = 1 - 5 \cdot 10^{(K-k+1)} \cdot \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j) \tag{300}$$

From equations 299 and 300 we get:

$$\mathscr{W}_v(w_i) = 1 - 5 \cdot 10^{(K-k+1)} \cdot \delta - \sum_{j=i+1}^{K} \mathscr{W}_v(w_j).$$

Rearranging the terms in the above inequality, we get:

$$\sum_{j=i}^{K} \mathscr{W}_v(w_j) = 1 - 5 \cdot 10^{(K-k+1)} \cdot \delta \tag{301}$$

Since $\mathscr{W}_v(w) \geq \sum_{j=i}^{K} \mathscr{W}_v(w_j)$, equation 301 implies that:

$$\mathscr{W}_v(w) \geq 1 - 5 \cdot 10^{(K-k+1)} \cdot \delta \tag{302}$$

From equations 298 and 302 we get:

$$\mathscr{W}_v(w) + (4/3) \cdot b_i^r(v) \geq 1 + (1/3) \cdot 10^{(K-k+1)} \cdot \delta \geq 1 + (\delta/3) \tag{303}$$

The last inequality holds since $2 \leq k \leq K$. The claim follows from equation 303 in this case.

□

10 The algorithm

In this section, we will present a dynamic algorithm for maintaining a $(1+\varepsilon)^2$-approximation to the size of the fractional matching $(w + w^r + w_1^*)$. First, we recall a standard assumption used in dynamic algorithms literature on the sequence of edge insertions/deletions in the input graph.

Assumption 10.1. *The input graph $G = (V,E)$ is empty (i.e., $E = \emptyset$) in the beginning of the sequence of edge insertions/deletions.*

However, we will make an (apparently) stronger assumption on the input sequence, as stated below.

Assumption 10.2. *The input graph $G = (V,E)$ is empty (i.e., $E = \emptyset$) in the beginning of the sequence of edge insertions/deletions. Further, the input graph $G = (V,E)$ is also empty (i.e., $E = \emptyset$) at the end of the sequence of edge insertions/deletions.*

In the theorem below, we show that Assumption 10.2 is without any loss of generality.

Theorem 10.3. *Suppose that a dynamic algorithm has an amortized update time of $\kappa(n)$, where $n = |V|$ is the number of nodes in the input graph, under Assumption 10.2. Then the same dynamic algorithm has an amortized update time of $O(\kappa(n))$ under Assumption 10.1.*

Proof. Consider a dynamic algorithm that has an amortized update time of $\kappa(n)$ under Assumption 10.2. Now, consider a sequence of t edge insertions/deletions that satisfy Assumption 10.1. Hence, the graph G is empty in the beginning of this sequence. Let $G^{(t)} = (V,E^{(t)})$ denote the status of the input graph at the end of this sequence. Note that $G^{(t)}$ need not be empty. Define $|E^{(t)}| = m^{(t)}$, and note that $m^{(t)} \leq t$.

We now delete all the edges from $G^{(t)}$ one after the other. This leads to a new sequence of $(t + m^{(t)})$ edge insertions/deletions where the input graph is empty in the end. In other words, the new sequence satisfies Assumption 10.2. Hence, the total time spent by the algorithm under this new sequence is at most $(t + m^{(t)}) \cdot \kappa(n) \leq 2t \cdot \kappa(n) = O(t \cdot \kappa(n))$. Since the old sequence is a prefix of the new sequence, we infer that the total time spent by the algorithm under the old sequence is at most the time spent under the new sequence. Thus, the total time spent under the old sequence (which consisted of only t edge insertions/deletions) is $O(t \cdot \kappa(n))$. So the amortized update time under the old sequence is $O(\kappa(n))$. This concludes the proof. □

Before proceeding any further, we recall the properties derived in Section 9.6, as they will be crucial in analyzing our algorithm. We also define the phrase "w-structures for a level $i \in \{2,\ldots,K\}$".

Definition 10.4. *Consider any level $i \in \{2,\ldots,K\}$. The phrase "w-structures for level i" refers to: The subgraph $G_i = (Z_i, E_i)$, the normal-capacities $\{b_i(x)\}, x \in Z_i$, the fractional assignment w_i with support $H_i \subseteq E_i$, and the node-weights $\{\mathscr{W}_x(w_i), W_x(w_i)\}$, $x \in Z_i$.*

From the invariants in Section 9.2, it is apparent that the w-structures for a level $i \in \{2,\ldots,K\}$ depend *only on* the w-structures for levels $j \in \{i+1,\ldots,K\}$. In particular, the w-structures for a level i do not in any way depend on the residual fractional assignment w^r (see equation 171) or the fractional assignment w_1^*. For the time being, we will focus on designing an algorithm that only maintains the w-structures for levels $\{2,\ldots,K\}$. This will immediately give us the size of the fractional assignment $w = \sum_{j=2}^{K} w_j$. Later on we will show how to extend our algorithm to maintain good approximations to the sizes of the fractional assignments w^r and w_1^* as well. The rest of this section is organized as follows.

1. In Section 10.1, we present the data structures that will be used by our algorithm to maintain the w-structures for levels $\{2,\ldots,K\}$.

2. In Section 10.2, we show how to update the above data structures after an edge insertion/deletion in G. This gives us a dynamic algorithm for maintaining the w-structures for levels $\{2,\ldots,K\}$.
3. In Section 10.2.1, we analyze the amortized update time of our algorithm from Section 10.2. See Theorem 10.10.
4. In Section 10.4, we show how to maintain the size of the fractional matching w_1^*. See Theorem 10.30.
5. Finally, in Section 10.5, we show how to maintain a $(1+\varepsilon)^2$-approximation to the size of the residual fractional matching w^r. See Theorem 10.35.

Our main result is summarized in the theorem below.

Theorem 10.5. *We can maintain a $(1+\varepsilon)^2$-approximation to the size of the fractional matching $(w + w^r + w_1^*)$ in $O((10/\varepsilon)^{K+8} \cdot n^{2/K})$ update time.*

Proof. The theorem follows if we sum over the update times given in Theorems 10.10, 10.30, 10.35, and then apply equation 169. □

10.1 Data structures

Our algorithm keeps the following data structures.

- The input graph $G = (V, E)$ using adjacency lists.

- For every level $i \in \{2,\ldots,K\}$;

  - The subgraph $G_i = (Z_i, E_i)$ using adjacency lists.

  - For every node $v \in Z_i$, the capacity $b_i(v)$, and the weights $\mathscr{W}_v(w_i)$, $W_v(w_i)$. We "extend" the capacities $\{b_i(v)\}$ and the node-weights $\{\mathscr{W}_v(w_i), W_v(w_i)\}$ in a natural to the input graph $G = (V, E)$. Thus, for every node $v \in V \setminus Z_i$, we set $b_i(v) = \mathscr{W}_v(w_i) = W_v(w_i) = 0$.

  - The support $H_i = \{e \in E_i : w_i(e)\}$ of the fractional assignment w_i using a doubly linked list. We assume that each edge $e \in H_i$ gets a weight $w_i(e) = 1/d_i = 1/n^{(i-1)/K}$, without *explicitly storing*

these edge-weights anywhere in our data structures.

- The subgraph (Z_i, H_i) using adjacency lists.

- The partition of the node-set Z_i into subsets $T_i, B_i, S_i \subseteq Z_i$. The sets T_i, B_i, S_i are maintained as doubly linked lists. Given any node $v \in Z_i$, the data structure can report in constant time whether the node is tight (i.e., $v \in T_i$) or big (i.e., $v \in B_i$) or small (i.e., $v \in S_i$).

- Whenever a node (resp. an edge) appears in a doubly linked list described above, we store a pointer from that node (resp. edge) to its position in the linked list. Using these pointers, we can insert (resp. delete) any element into (resp. from) any list in constant time.

10.2 Handling the insertion/deletion of an edge (u,v) in the input graph $G = (V,E)$

Suppose that an edge (u,v) is either inserted into or deleted from G. To handle this edge insertion/deletion, we first update the adjacency lists in $G = (V,E)$. Next, we update the w-structures for the levels in a "top down" manner, as described in Figure 6. The set L_j consists of the subset of nodes $x \in V$ whose discretized weight $\mathscr{W}_x(w_j)$ changes its value while fixing the w-structures for level i as per Section 10.2.1.

```
1.    For i = K to 2;
2.        Set L_i ← ∅.
3.        Call the subroutine FIX-STRUCTURES(i). See Section 10.2.1.
```

Figure 6

In Section 10.2.1, whenever in set H_i we insert/delete an edge incident upon a node z, we will call the subroutine UPDATE(i,z) described below. This subroutine updates the value of $\mathscr{W}_z(w_i)$ *in a lazy manner* so as to ensure that Invariant 9.10 is satisfied.

To be more specific, just before changing the value of $W_z(w_i)$, Invariant 9.10 was satisfied. Now, an edge insertion/deletion changes the value of $W_z(w_i)$ by $1/d_i < \varepsilon$ (see Definition 9.8, Invariant 9.9 and equation 166). Hence, we need to change the value of $\mathscr{W}_z(w_i)$ by at most ε. This is done in Figure 7 in a lazy manner.

1. If the current value of $\mathscr{W}_z(w_i)$ is too small to satisfy Invariant 9.10, then we increase $\mathscr{W}_z(w_i)$ by ε.
2. Else if the current value of $\mathscr{W}_z(w_i)$ is too large to satisfy Invariant 9.10, then we decrease $\mathscr{W}_z(w_i)$ by ε.

```
1.    If 𝒲_v(w_i) < W_v(w_i), then
2.        Set 𝒲_v(w_i) ← 𝒲_v(w_i) + ε.
3.    Else if W_v(w_i) ≤ 𝒲_v(w_i) − 2ε, then
4.        Set 𝒲_v(w_i) ← 𝒲_v(w_i) − ε.
```

Figure 7: UPDATE(i,z).

10.2.1 The subroutine FIX-STRUCTURES(i), where $i \in \{2,\ldots,K\}$.

Consider the scenario where we have fixed the w-structures for all levels $j > i$. At this stage, for $j \in \{i+1,\ldots,K\}$, the set L_j consists of all the nodes $x \in V$ such that the value of $\mathscr{W}_x(w_j)$ has been changed while fixing the w-structures for level j. Further, at this stage we have $L_j = \emptyset$ for all $j \in \{2,\ldots,i\}$. The procedure in this section will serve two objectives: (a) it will update the w-structures for level i, and (b)

whenever the value of $\mathscr{W}_x(w_i)$ is changed for any node x, it will set $L_i \leftarrow L_i \cup \{x\}$. Thus, at any point in time, the set L_i will consist of exactly those nodes whose $\mathscr{W}_x(w_i)$ values have been changed till now. Note that to fix the w-structures for level i, we first have to update the subgraph $G_i = (Z_i, E_i)$ and the support $H_i \subseteq E_i$ of the fractional assignment w_i. Specifically, we have to address two types of issues if $i < K$, as described below.

1. Since we have already fixed the w-structures for levels $j > i$, it is plausible that there has been a change in the node-set Z_{i+1} and the partition of Z_{i+1} into subsets $T_{i+1}, B_{i+1}, S_{i+1}$. To satisfy Definition 9.6, we might have to insert (resp. delete) some nodes into (resp. from) the set Z_i.
   - (a) According to Definition 9.6 and Lemma 9.26, whenever we insert a node x into the subset Z_i (this happens if $x \in Z_{i+1} \setminus T_{i+1}$ and $x \notin Z_i$), we have to ensure that for all nodes $y \in Z_i$ with $(x,y) \in E_i \subseteq H_{i+1}$, the edge (x,y) is inserted into E_i. Accordingly, we have to scan through all the edges $(x,y) \in H_{i+1}$, and for every such edge (x,y), if we find that $y \in Z_i$, then we need to insert the edge (x,y) into the subgraph $G_i = (Z_i, E_i)$.
   - (b) According to Definition 9.6, whenever we delete a node x from the subset Z_i (this happens if $x \in Z_i$ and $x \notin Z_{i+1} \setminus T_{i+1}$), we also have to delete all its incident edges $(x,y) \in E_i$ from the subgraph $G_i = (Z_i, E_i)$. Further, when deleting an edge (x,y) from E_i, we have to check if $(x,y) \in H_i$, and if the answer is yes, then we have to delete the edge (x,y) from the set H_i as well (see Invariant 9.9). The last step will decrease each of the weights $W_x(w_i), W_y(w_i)$ by $1/d_i$ (since $w_i(e) = 1/d_i$ for all $e \in H_i$), and hence it might also change the values of $\mathscr{W}_x(w_i)$ and $\mathscr{W}_y(w_i)$, which, in turn, might lead us to reassign y in the partition of the node-set Z_i into subsets T_i, S_i, B_i.

2. There might be some node $x \in V$ that remains in the set Z_i, but whose normal-capacity $b_i(x)$ has to be changed (for we have already changed the w-structures for levels $j > i$, and they determine the value of $b_i(x)$ as per Definition 9.7). In this event, we might encounter a situation where the value of $\mathscr{W}_x(w_i)$ exceeds the actual value of $b_i(x)$. To fix this issue, for such nodes x, as a precautionary measure we "turn off" all its incident edges $(x,y) \in H_i$ with nonzero weights under w_i. Specifically, we visit every edge $(x,y) \in H_i$, and delete the edge (x,y) from H_i. This last step decreases the weights $W_x(w_i), W_y(w_i)$ by $1/d_i$, and accordingly, we might need to change the values of $\mathscr{W}_x(w_i)$ and $\mathscr{W}_y(w_i)$ as well. As a result, we might need to reassign the nodes x, y in the partition of Z_i into subsets T_i, S_i, B_i.

Let us say that a "bad event" happens for a node x at level i iff either the node gets inserted into (resp. deleted from) the set Z_i or its normal-capacity $b_i(x)$ gets changed. From Definitions 9.7, 9.11, 9.6 and equation 172, we conclude that such a bad event happens only if the discretized weight $\mathscr{W}_x(w_j)$ of the node at some level $j > i$ gets changed. In other words, a bad event happens for a node x at level i only if $x \in \bigcup_{j=i+1}^{K} L_j$. It follows that all the operations described above are implemented in Figure 8. To be more specific, Steps (02) – (07) deal with Case 1(a), Steps (08) – (19) deal with Case 1(b), and Steps (20) – (29) deal with Case 2. Finally, we note that if $i = K$, then we do not need to worry about any of these issues since then $\bigcup_{j=i+1}^{K} L_j = \emptyset$. The node-set Z_K and the capacities $\{b_K(x)\}$ remain unchanged (see Definitions 9.6, 9.7).

We now analyze the total time it takes to perform the operations in Figure 8. The main For loop in line 01 runs for $\left|\bigcup_{j>i} L_j\right|$ iterations. During each such iteration, it takes time proportional to either $\deg_x(H_{i+1})$ (see the For loop in line 06) or $\deg_x(E_i)$ (see the For loops in lines 10, 22). Since $d_{i+1} = d_i \cdot n^{1/K}$ (see Definition 9.8), Lemma 9.27 and Corollary 9.28 imply that $\deg_x(H_{i+1}) \leq d_i \cdot n^{1/K}$ and $\deg_x(E_i) \leq d_i \cdot n^{1/K}$. Hence, the time taken by each iteration of the main For loop is $O(d_i \cdot n^{1/K})$. Accordingly, the total runtime becomes $O(\left|\bigcup_{j>i} L_j\right| \cdot d_i \cdot n^{1/K})$, as stated in the lemma below.

Lemma 10.6. *It takes $O\left(\left|\bigcup_{j=i+1}^{K} L_j\right| \cdot d_i \cdot n^{1/K}\right)$ time to perform the operations in Figure 8.*

Remark. Note that for $i = K$ the set $\bigcup_{j=i+1}^{K} L_j$ is empty. So we do not execute any of the steps in Figure 8.

01. For all nodes $x \in \bigcup_{j=i+1}^{K} L_j$;
02. If $x \in Z_{i+1} \setminus T_{i+1}$ and $x \notin Z_i$, then *(Definition 9.6 is violated)*
03. Insert the node x into the subsets Z_i and S_i.
04. Update the value of $b_i(x)$, based on the structures for levels $j > i$ *(see Definition 9.7)*.
05. Set $\mathscr{W}_x(w_i) = \varepsilon$, $W_x(w_i) = 0$. *(Note that $\mathscr{W}_x(w_i) = \varepsilon \leq b_i(x)$ by Observation 9.24.)*
06. For every edge $(x,y) \in H_{i+1}$;
07. If $y \in Z_i$, then insert the edge (x,y) into the subgraph $G_i = (Z_i, E_i)$.
08. Else if $x \notin Z_{i+1} \setminus T_{i+1}$ and $x \in Z_i$, then *(Definition 9.6 is violated)*
09. Delete the node x from the subset Z_i and from $T_i \cup S_i \cup B_i$.
10. For every edge $(x,y) \in E_i$;
11. Delete the edge (x,y) from the subgraph $G_i = (Z_i, E_i)$.
12. If $(x,y) \in H_i$, then
13. Delete the edge (x,y) from the subset H_i.
14. For each node $z \in \{x,y\}$;
15. Decrease the value of $W_z(w_i)$ by $1/d_i$.
16. Update the value of $\mathscr{W}_z(w_i)$ by calling the subroutine UPDATE(i, z_i).
17. If the value of $\mathscr{W}_z(w_i)$ changes in the previous step, then set $L_i \leftarrow L_i \cup \{z\}$.
18. Reassign y to one of the subsets T_i, S_i, B_i according to Definition 9.11.
19. Set $\mathscr{W}_x(w_i) = b_i(x) = 0$.
20. Else if $x \in Z_{i+1} \setminus T_{i+1}$ and $x \in Z_i$, then *(the value of $b_i(x)$ might have changed)*
21. Update the value of $b_i(x)$, based on the structures for levels $j > i$ *(see Definition 9.7)*.
22. For every edge $(x,y) \in E_i$;
23. If $(x,y) \in H_i$, then
24. Delete the edge (x,y) from the subset H_i.
25. For each $z \in \{x,y\}$;
26. Decrease $W_z(w_i)$ by $1/d_i$.
27. Update the value of $\mathscr{W}_z(w_i)$ by calling the subroutine UPDATE(i, z).
28. If the value of $\mathscr{W}_z(w_i)$ changes in the previous step, then set $L_i \leftarrow L_i \cup \{z\}$.
29. Reassign z to one of the subsets T_i, S_i, B_i according to Definition 9.11.

Figure 8: Updating the subgraph $G_i = (Z_i, E_i)$.

We have not yet updated the adjacency list data structures of $G_i = (Z_i, E_i)$ to reflect the insertion/deletion of the edge (u,v). This is done in Figure 9. We only need to update the data structures if both the endpoints u, v belong to Z_i, for otherwise by Definition 9.6 the edge (u,v) does not participate in the subgraph $G_i = (Z_i, E_i)$. Note that if the edge (u,v) is to inserted into G_i (lines 01-02 in Figure 9), then we leave the edge-set H_i unchanged. Else if the edge is to be deleted from G_i (lines 03-11 in Figure 9), then we first delete it from E_i, and then check if the edge also belonged to H_i. If the answer is yes, then we delete (u,v) from H_i as well. Thus, the weights $W_u(w_i)$ and $W_v(w_i)$ decrease by $1/d_i$ due to this operation. Accordingly, the discretized weights $\mathscr{W}_u(w_i)$ and $\mathscr{W}_v(w_i)$ might also undergo some changes. If any of these discretized weights does get modified, then we insert the corresponding node into L_i. All these operations take constant time.

Lemma 10.7. *It takes $O(1)$ time to perform the operations in Figure 9.*

```
01. If u, v ∈ Z_i and the edge (u,v) has been inserted into G = (V,E), then
02.     Insert the edge (u,v) into the subgraph G_i = (Z_i, E_i).
03. Else if u, v ∈ Z_i and the edge (u,v) has been deleted from G = (V,E), then
04.     Delete the edge (u,v) from the subgraph G_i = (Z_i, E_i).
05.     If (u,v) ∈ H_i, then
06.         Delete the edge (u,v) from the subset H_i.
07.         For each node x ∈ {u,v};
08.             Decrease the value of W_x(w_i) by 1/d_i.
09.             Update the value of 𝒲_x(w_i) by calling the subroutine UPDATE(i,z).
10.             If the value of 𝒲_x(w_i) changes in the previous step, then set L_i ← L_i ∪ {x}.
11.             Reassign the node x into one of the subsets T_i, S_i, B_i according to Definition 9.11.
```

Figure 9: Inserting/deleting the edge (u,v) in $G_i = (Z_i, E_i)$.

By this time, we have fixed the adjacency lists for the subgraph $G_i = (Z_i, E_i)$, and we have also ensured that $\mathscr{W}_x(w_i) \leq b_i(x)$ for every node $x \in Z_i$. However, due to the deletions of multiple edges from H_i, we might end up in a situation where we find two nodes $x, y \in Z_i \setminus T_i$ connected by an edge $(x,y) \in E_i$, but the edge (x,y) is not part of H_i (thereby violating Invariant 9.12). This can happen only if at least one of the nodes x, y are part of $\bigcup_{j=i}^{K} L_j$. For if both $x, y \notin \bigcup_{j=i}^{K} L_j$, then it means that there have been no changes in their normal capacities $\{b_j(z)\}_{z \in \{x,y\}}$ and discretized weights $\{\mathscr{W}_z(w_j)\}_{z \in \{x,y\}}$ at levels $j \geq i$. It follows that both x, y belonged to $Z_i \setminus T_i$ just prior to the insertion/deletion of (u,v) in G, and hence the edge (x,y) was also part of H_i at that instant (to satisfy Invariant 9.12). Further, the edge (x,y) was also not deleted from H_i in Figure 8, for otherwise either x or y would have been part of $\bigcup_{j=i+1}^{K} L_j$. We conclude that if both $x, y \notin \bigcup_{j \geq i} L_j$ and $x, y \in Z_i \setminus T_i$, then the edge (x,y) must belong to H_i at this moment.

Accordingly, we perform the operations in Figure 10, whereby we go through the list of nodes in $\bigcup_{j=i}^{K} L_j$, and for each node x in this list, we check if $x \in Z_i$. If the answer is yes, then we visit all the edges $(x,y) \in E_i \setminus H_i$ incident upon x. If we find that both x, y do not belong to T_i, then to satisfy Invariant 9.12 we insert the edge (x,y) into the set H_i. The time taken for these operations is $\left|\bigcup_{j \geq i} L_j\right|$ times the maximum degree of a node in E_i, which is $d_i \cdot n^{1/K}$ by Corollary 9.28. Thus, we get the following lemma.

Lemma 10.8. *The time taken for the operations in Figure 10 is $O\left(\left|\bigcup_{j=i}^{K} L_j\right| \cdot d_i \cdot n^{1/K}\right)$.*

At this stage, we have fixed the w-structures for level i, and hence we finish the execution of the subroutine FIX-STRUCTURES(i). We conclude this section by deriving a bound on the running time of this subroutine. This bound follows from Lemmas 10.6, 10.7 and 10.8.

Lemma 10.9. *The subroutine FIX-STRUCTURES(i) runs for $O\left(1 + \left|\bigcup_{j=i}^{K} L_j\right| \cdot d_i \cdot n^{1/K}\right)$ time.*

1. Set $L \leftarrow \bigcup_{j=i}^{K} L_j$.
2. For each node $x \in L$;
3. If $x \in Z_i$, then call the subroutine FIX(i,x). See Figure 11.

Figure 10: Handling the nodes rendered non-tight so far.

1. For each edge $(x,y) \in E_i$;
2. If $(x,y) \notin H_i$ and $x \notin T_i$ and $y \notin T_i$, then
3. Insert the edge (x,y) to the subset H_i.
4. For each node $z \in \{x,y\}$;
5. Increase the value of $W_z(w_i)$ by $1/d_i$.
6. Update the value of $\mathscr{W}_z(w_i)$ by calling the subroutine UPDATE(i,z).
7. If the value of $\mathscr{W}_z(w_i)$ changes in the previous step, then set $L_i \leftarrow L_i \cup \{z\}$.
8. Reassign the node z into one of the subsets T_i, S_i, B_i according to Invariant 9.12.

Figure 11: FIX(i,x).

10.3 Analyzing the amortized update time

We analyze the amortized update time of our algorithm (as described in Section 10.2) over a sequence of edge insertions/deletions in the input graph $G = (V,E)$. We prove an amortized update time of $O((10/\varepsilon)^{K+2} \cdot n^{1/K})$. Since the algorithm in Section 10.2 maintains the w-structures (see Definition 10.4) for all the levels $j \in \{2,\ldots,K\}$, we can easily augment it to maintain the size of the fractional matching $w = \sum_{j=2}^{K} w_j$ without incurring any additional overhead in the update time. This leads to the following theorem.

Theorem 10.10. *We can maintain the size of w in $O((10/\varepsilon)^{K+2} \cdot n^{1/K})$ amortized update time.*

Following Assumption 10.2 and Theorem 10.3, we assume that the input graph is empty (i.e., $E = \emptyset$) in the beginning *and* at the end of this sequence of edge insertions/deletions. We use a charging scheme to analyze the amortized update time. Specifically, we create the following "bank-accounts".

- For each node $v \in V$ and level $i \in \{2,\ldots,K\}$, there are two accounts, namely:
  - NORMAL-ACCOUNT$[v,i]$.
  - WORK-ACCOUNT$[v,i]$.
- For level $i = 1$, each node $v \in V$ has exactly one account, namely:
  - WORK-ACCOUNT$[v,1]$.
- For every unordered pair of nodes (u,v) and level $i \in \{1,\ldots,K\}$, there is one bank account, namely:
  - WORK-ACCOUNT$[(u,v),i]$.
- From this point onwards, the phrase "*work-account*" will refer to any bank account of the form WORK-ACCOUNT$[v,i]$ or WORK-ACCOUNT$[(u,v),i]$. Similarly, the phrase "*normal-account*" will refer to any bank account of the form NORMAL-ACCOUNT$[v,i]$.

While handling an edge insertion/deletion in G, we will sometimes "transfer" money from one bank account to another. Further, we will allow the bank accounts to have negative balance. For example, if an account has x dollars, and we transfer $y > x$ dollars from it to some other account, the balance in the first account will become $(x-y)$ dollars, which is a negative amount. We will ensure that the following properties hold.

Property 10.11. *In the beginning (before the first edge insertion in G), each bank account has zero balance.*

Property 10.12. *For each edge insertion/deletion in the input graph G, we deposit $O((10/\varepsilon)^{K+2} \cdot n^{1/K})$ dollars into bank-accounts.*

Property 10.13. *Money is never withdrawn from a work-account. So it always has a nonnegative balance.*

Property 10.14. *In the end (when G becomes empty again), each normal-account has a nonnegative balance.*

Property 10.15. *The total balance in the work-accounts is at least the total update time.*

Properties 10.13, 10.14 and 10.15 imply that at the end of the sequence of edge insertions/deletions, the sum of the balances in the bank accounts is at least the total update time of the algorithm. By Properties 10.11 and 10.12, the former quantity is $O(10/\varepsilon)^{K+2} \cdot n^{1/K})$ times the number of edge insertions/deletions in G. Hence, we get an amortized update time of $O((10/\varepsilon)^{K+2} \cdot n^{1/K})$. We now specify the exact rules that govern the functioning of all the bank accounts. We will show that these rules satisfy all the properties described above. This will conclude the proof of the amortized update time.

10.3.1 Rules governing the bank accounts

The first rule describes the initial conditions.

Rule 10.16. *In the beginning (before the first edge insertion in G), every bank account has zero balance.*

The next rule states how to deposit money into the bank accounts after an edge insertion/deletion in G.

Rule 10.17. *When an edge (u,v) is inserted into (resp. deleted from) G, we make the following deposits.*

- *For each $i \in \{1,\ldots,K\}$;*
  - *We deposit one dollar into the account* WORK-ACCOUNT$[(u,v),i]$.
- *For each $i \in \{2,\ldots,K\}$;*
  - *We deposit $(10/\varepsilon)^{i+1} \cdot n^{1/K}$ dollars into* NORMAL-ACCOUNT$[x,i]$ *for all $x \in \{u,v\}$.*

Before proceeding any further, we need to make a simple observation. Consider any level $i \in \{2,\ldots,K\}$, and the set of edges H_i that form the support of the fractional assignment w_i. Our algorithm in Section 10.2 deletes an edge $(x,y) \in H_i$ from the set H_i due to one of the following reasons:

- The edge (x,y) is getting deleted from the input graph $G = (V,E)$, and hence we have to delete the edge from H_i (see Steps 04, 06 in Figure 9). We say that this is a "*natural deletion*" from the set H_i.
- Some other edge (x',y') is getting deleted from the input graph $G = (V,E)$. While handling this edge deletion, we have to fix the w-structures for level i, and as a result the edge (x,y) gets deleted from H_i (see Steps 13, 24 in Figure 8). In this event, the edge (x,y) does *not* get deleted from the input graph G itself. We say that this is an "*artificial deletion*" from the set H_i.

From our algorithm in Section 10.2, we make the following observation.

Observation 10.18. *No artificial deletion takes place from the edge-set H_K. Further, at any level $2 \leq i < K$, an edge (x,y) gets artificially deleted from the set H_i only when it has at least one endpoint in $\bigcup_{j>i} L_j$.*

We now describe the rule that governs the transfer of money from one normal account to another.

Rule 10.19. *Consider any level $2 \leq i \leq K$. By Observation 10.18, an edge (x,y) gets artificially deleted from H_i only if $i < K$, and when such an event occurs, the edge must have at least one endpoint (say x) in L_j for some $i < j \leq K$. At that instant, for all $z \in \{x,y\}$, we transfer $(10/\varepsilon)^{i+1} \cdot n^{1/K}$ dollars from* NORMAL-ACCOUNT$[x,j]$ *to* NORMAL-ACCOUNT$[z,i]$.

Finally, we state the rule governing the transactions between the normal-accounts and the work-accounts.

Rule 10.20. *Suppose that while handling an edge insertion/deletion in G some node $x \in V$ becomes part of the set L_i. At that instant, for each $1 \leq j \leq i$, we transfer $d_j \cdot n^{1/K}$ dollars from* NORMAL-ACCOUNT$[x,i]$ *to* WORK-ACCOUNT$[x,j]$.

Rule 10.16 ensures that Property 10.11 is satisfied. According to Rule 10.17, when an edge insertion/deletion takes places in G, the total amount of money deposited to the bank accounts is at most:

$$K + \sum_{i=2}^{K} (10/\varepsilon)^{i+1} \cdot n^{1/K} = K + O((10/\varepsilon)^{K+2} \cdot n^{1/K}) = O((10/\varepsilon)^{K+2} \cdot n^{1/K}).$$

The last equality holds since $K < n^{1/K}$ (see equation 167). Thus, Property 10.12 is also satisfied. We also note that the four rules described above immediately imply Property 10.13. Next, we focus on proving Property 10.15. Towards this end, we focus on any given edge insertion/deletion in G. We handle this event as per the procedure in Figure 6. Thus, according to Lemma 10.9, the total time taken to handle this edge insertion/deletion in G is given by:

$$\sum_{i=2}^{K} \left(1 + \left| \bigcup_{j=i}^{K} L_j \right| \cdot d_i \cdot n^{1/K} \right) \tag{304}$$

Due to this edge insertion/deletion in G, as per Rule 10.17 the sum $\sum_{(x,y)} \sum_{i=1}^{K}$ WORK-ACCOUNT$[(x,y),i]$ increases by:

$$\sum_{i=1}^{K} 1 \tag{305}$$

According to Rule 10.20, for every level $i \in \{1,\ldots,K\}$, the sum $\sum_{x \in V}$ WORK-ACCOUNT$[x,i]$ increases by:

$$\left| \bigcup_{j=i}^{K} L_j \right| \cdot d_i \cdot n^{1/K} \tag{306}$$

Thus, due the edge insertion/deletion in G, the sum of all the work-accounts increases by at least:

$$\sum_{i=1}^{K} \left(1 + \left| \bigcup_{j=i}^{K} L_j \right| \cdot d_i \cdot n^{1/K} \right) \tag{307}$$

From equations 304 and 307, we reach the following conclusion.

- Due to each edge insertion/deletion in G, the total balance in the work-accounts increases by an amount that is at least the total time spent to handle that edge insertion/deletion.

The above statement, along with Property 10.11 and 10.13 (which we have proved already), implies Property 10.15. Thus, it remains to prove Property 10.14, which is done by the lemma below. Its proof appears in Section 10.3.2. As there is no normal-account at level one, the lemma takes care of all the normal-accounts. This concludes the proof of Theorem 10.10.

Lemma 10.21. *Consider any node $x \in V$ and any level $i \in \{2,\ldots,K\}$. At the end of the sequence of edge insertions/deletions (when G becomes empty), we have a nonnegative balance in* NORMAL-ACCOUNT$[x,i]$.

10.3.2 Proof of Lemma 10.21

Throughout this section, we will use the phrase "time-horizon" to refer to the time-interval that begins just before the first edge insertion in G (when G is empty), and ends after the last edge deletion from G (when G becomes empty again). We will also use the following notations.

- Let Λ^+ be the total amount of money deposited into NORMAL-ACCOUNT$[x,i]$ during the entire time-horizon. Similarly, let Λ^- be the total amount of money withdrawn from NORMAL-ACCOUNT$[x,i]$ during the entire time-horizon. We will show that $\Lambda^+ \geq \Lambda^-$. Since NORMAL-ACCOUNT$[x,i]$ has zero balance before the first edge insertion in G (see Rule 10.16), this will imply Lemma 10.21.

- Consider any given edge insertion/deletion in G. We say that this edge insertion/deletion is "*critical*" iff the following property holds: While handling the edge insertion/deletion in G using the algorithm from Section 10.2, the node x becomes part of the set L_i.

  Let $\Delta[x,i]$ be the total number of *critical* edge insertions/deletions in G during the time-horizon.

- Let $m^-[x,i]$ (resp. $m^+[x,i]$) denote the total number of times an edge incident upon x is deleted from (resp. inserted into) the set H_i during the entire time-horizon. Since the input graph G is empty both before and after the time-horizon, we have $m^+[x,i] = m^-[x,i]$.

Roadmap for the proof. In Claim 10.22, we lower bound Λ^+ in terms of $m^-[x,i]$. In Claim 10.23, we upper bound Λ^- in terms of $\Delta[x,i]$. In Claim 10.24, we related the two quantities $\Delta[x,i]$ and $m^-[x,i]$. Next, in Corollary 10.25, we use Claims 10.23 and 10.24 to upper bound Λ^- in terms of $m^-[x,i]$. Finally, Claim 10.22 and Corollary 10.25 imply that $\Lambda^+ \geq \Lambda^-$. This concludes the proof of the lemma.

Claim 10.22. *We have:* $\Lambda^+ \geq m^-[x,i] \cdot (10/\varepsilon)^{i+1} \cdot n^{1/K}$.

Proof. Rule 10.17 and 10.19 implies that each time an edge incident upon x gets deleted from H_i, we deposit $(10/\varepsilon)^{i+1} \cdot n^{1/K}$ dollars into NORMAL-ACCOUNT$[x,i]$. Since $m^-[x,i]$ denotes the total number of edge deletions incident upon x that take place in H_i, the claim follows. □

Claim 10.23. *We have:* $\Lambda^- \leq \Delta[x,i] \cdot 5 \cdot (10/\varepsilon)^i \cdot d_i \cdot n^{1/K}$.

Proof. Consider an edge insertion/deletion in the input graph. Suppose that the node x becomes part of the set L_i while we handle this edge insertion/deletion using the algorithm from Section 10.2. In other words, this edge insertion/deletion in G contributes one towards the value of $\Delta[x,i]$. We will show that while handling this edge insertion/deletion, we withdraw at most $5 \cdot (10/\varepsilon)^i \cdot d_i \cdot n^{1/K}$ dollars from NORMAL-ACCOUNT$[x,i]$.

We first bound the total amount of money that are withdrawn from NORMAL-ACCOUNT$[x,i]$ due to Rule 10.19. Consider any level $j \in \{2,\ldots,i-1\}$. While handling the edge insertion/deletion in G, the algorithm from Section 10.2 deletes at most $\deg_x(H_j)$ edges incident upon x from the set H_j. For each such deletion of an edge (x,y), we withdraw $2 \cdot (10/\varepsilon)^{j+1} \cdot n^{1/K}$ dollars from NORMAL-ACCOUNT$[x,i]$, and distribute this amount evenly between NORMAL-ACCOUNT$[x,j]$ and NORMAL-ACCOUNT$[y,j]$. Hence, the total amount of money withdrawn from NORMAL-ACCOUNT$[x,i]$ due to Rule 10.19 is at most:

$$\sum_{j=2}^{i-1} \deg_x(H_j) \cdot 2 \cdot (10/\varepsilon)^{j+1} \cdot n^{1/K} \leq \sum_{j=2}^{i-1} d_j \cdot 2 \cdot (10/\varepsilon)^{j+1} \cdot n^{1/K} \tag{308}$$

$$\leq \sum_{j=2}^{i-1} d_i \cdot 2 \cdot (10/\varepsilon)^{j+1} \cdot n^{1/K} \tag{309}$$

$$\leq d_i \cdot 4 \cdot (10/\varepsilon)^i \cdot n^{1/K} \tag{310}$$

Equation 308 follows from Lemma 9.27. Equation 309 holds since $d_j \leq d_i$ for all $j \leq i$ (see Definition 9.8).

Next, we bound the total amount of money withdrawn from NORMAL-ACCOUNT$[x,i]$ due to Rule 10.20. For each level $j \in \{1,\ldots,i\}$, we withdraw $d_j \cdot n^{1/K}$ dollars from NORMAL-ACCOUNT$[x,i]$ and transfer this amount to WORK-ACCOUNT$[x,j]$. Hence, the total amount withdrawn from NORMAL-ACCOUNT$[x,i]$ due to Rule 10.20 is given by:

$$\sum_{j=1}^{i} d_j \cdot n^{1/K} \leq 2 \cdot d_i \cdot n^{1/K} \tag{311}$$

$$\leq (10/\varepsilon)^i \cdot d_i \cdot n^{1/K} \tag{312}$$

Equation 311 follows from Definition 9.8 and equation 168. Finally, we note that Rules 10.20 and 10.19 are the only rules that govern the withdrawal of money from NORMAL-ACCOUNT$[x,i]$. Hence, adding equations 310 and 312, we reach the following conclusion.

- At most $5 \cdot (10/\varepsilon)^i \cdot d_i \cdot n^{1/K}$ are withdrawn from NORMAL-ACCOUNT$[x,i]$ while handling the insertion/deletion of an edge in G that results in x becoming part of L_i.

Since $\Delta[x,i]$ is the number of times x becomes part of L_i during the entire time-horizon, the claim follows. □

Claim 10.24. *We have:* $(\varepsilon d_i/2) \cdot \Delta[x,i] \leq m^-[x,i]$.

Proof. Recall the notion of a "*critical*" edge insertion/deletion in G. Such an edge insertion/deletion is characterized by the following property: While handling such an edge insertion/deletion using the algorithm from Section 10.2, the node x becomes part of the set L_i. Also recall that $\Delta[x,i]$ denotes the total number of critical edge insertions/deletions during the entire time-horizon.

Accorinngly, by definition, between any two critical edge insertions/deletions in G, the discretized weight $\mathscr{W}_x(w_i)$ (which is an integral multiple of ε) must have changed by at least ε, since the node x becomes part of L_i only when its discretized weight $\mathscr{W}_x(w_i)$ changes. Recall that the discretized weight $\mathscr{W}_x(w_i)$ is updated in a *lazy manner* after a change in the weight $W_x(w_i)$ (see Figure 7). Further, in one step the weight $W_x(w_i)$ changes by $1/d_i$ (since each edge in the support of w_i has weight $1/d_i$), and we have $1/d_i \leq 1/n^{1/K} \ll \varepsilon$ (see Definition 9.8 and equation 166). Accordingly, we infer that between any two critical edge insertions/deletions in G, the weight $W_x(w_i)$ also changes by at least ε. Next, note that the weight $W_x(w_i)$ changes by $1/d_i$ only when there is an edge insertion/deletion incident upon x in H_i. Hence, an ε change in the weight $W_x(w_i)$ corresponds to εd_i edge insertions/deletions in H_i incident upon x.

Let $m[x,i] = m^+[x,i] + m^-[x,i]$ denote the total number of edge insertions/deletions in H_i incident upon x during the entire time-horizon. The above discussion implies that $\Delta[x,i] \cdot (\varepsilon d_i) \leq m[x,i]$. By Assumption 10.2 (also see Theorem 10.3), we have $m^+[x,i] = m^-[x,i]$ and hence $m[x,i] = 2 \cdot m^-[x,i]$. Accordingly, we get: $(\varepsilon d_i) \cdot \Delta[x,i] \leq m[x,i] = 2 \cdot m^-[x,i]$. The claim follows. □

Corollary 10.25. *We have:* $\Lambda^- \leq m^-[x,i] \cdot (10/\varepsilon)^{i+1} \cdot n^{1/K}$.

Proof. From Claims 10.23 and 10.24, we infer that:

$$\Lambda^- \leq \Delta[x,i] \cdot 5 \cdot (10/\varepsilon)^i \cdot d_i \cdot n^{1/K} = (\varepsilon d_i/2) \cdot \Delta[x,i] \cdot (10/\varepsilon)^{i+1} \cdot n^{1/K} \leq m^-[x,i] \cdot (10/\varepsilon)^{i+1} \cdot n^{1/K}$$

□

From Claim 10.22 and Corollary 10.25, we infer that $\Lambda^+ \geq \Lambda^-$.

10.4 Maintaining the size of the fractional assignment w_1^*.

From Observations 9.23 and 9.24, we infer that for every node $v \in Z_1$ the capacity $b_1^*(v)$ is a positive integral multiple of ε. Further, Definition 9.20 ensures that $b_1^*(v) \leq 1$ for every node $v \in Z_1$. Thus, we get:

Observation 10.26. *For every node $v \in Z_1$, we have $b_1^*(v) = \kappa_v^* \cdot \varepsilon$ for some integer $\kappa_v^* \in [1, 1/\varepsilon]$.*

We now define a "meta-graph" $\mathscr{G}_1 = (\mathscr{Z}_1, \mathscr{E}_1)$. For clarity of exposition, we will refer to the nodes in $\mathscr{Z}_1$ as "meta-nodes" and the edges in $\mathscr{E}_1$ as "meta-edges". The meta-graph is constructed as follows. For each node $v \in Z_1$, we create κ_v^* meta-nodes $v(1), \ldots, v(\kappa_v^*)$, where κ_v^* is defined as in Observation 10.26. Next, for each edge $(u,v) \in E_1$, we create $\kappa_u^* \cdot \kappa_v^*$ meta-edges $\{(u(i), v(j))\}, 1 \leq i \leq \kappa_u^*, 1 \leq j \leq \kappa_v^*$. The next theorem bounds the size of the maximum matching in the meta-graph.

Theorem 10.27. *The size of the maximum (integral) matching in $\mathscr{G}_1 = (\mathscr{Z}_1, \mathscr{E}_1)$ equals $(1/\varepsilon)$ times the maximum size of a fractional b-matching in $G_1 = (Z_1, E_1)$ with respect to the node-capacities $\{b_1^*(v)\}, v \in Z_1$.*

Proof. (Sketch) The maximum possible size of a fractional b-matching in G_1 is given by the following LP.

$$\text{Maximize} \quad \sum_{(u,v) \in E_1} x^*(u,v) \tag{313}$$

$$\text{s. t.} \quad \sum_{(u,v) \in E_1} x^*(u,v) \leq \kappa_v^* \cdot \varepsilon \text{ for all nodes } v \in Z_1. \tag{314}$$

$$x^*(u,v) \geq 0 \text{ for all edges } (u,v) \in E_1^*. \tag{315}$$

Since the input graph $G = (V,E)$ is bipartite, the subgraph $G_1 = (Z_1, E_1)$ is also bipartite. Hence, the constraint matrix of the above LP is totally unimodular. Thus, in the optimal solution to the above LP, each variable $x^*(u,v)$ is going to take a value that is an integral multiple of ε. We can map such a solution $\{x^*(u,v)\}$ in a natural way to a matching $\mathscr{M}_1 \subseteq \mathscr{E}_1$ in the meta-graph: For every edge $(u,v) \in E$, include $x^*(u,v)/\varepsilon$ meta-edges from the collection $\{(u(i), v(j))\}, 1 \leq i \leq \kappa_u^*, 1 \leq j \leq \kappa_v^*$, into the matching $\mathscr{M}_1$. The size of the matching $\mathscr{M}_1$ will be exactly $(1/\varepsilon)$ times the LP-objective.

Similarly, given any matching $\mathscr{M}_1 \subseteq \mathscr{E}_1$ in the meta-graph, we can construct a feasible solution to the above LP in a natural way. Intially, set $x^*(u,v) = 0$ for all edges $(u,v) \in E_1$. Next, scan through the meta-edges in $\mathscr{M}_1$, and for every meta-edge of the form $(u(i), v(j)) \in \mathscr{M}_1$, set $x^*(u,v) \leftarrow x^*(u,v) + \varepsilon$. It is easy to check that at the end of this procedure, we will get a feasible solution to the above LP whose objective value is exactly ε times the size of $\mathscr{M}_1$. The theorem follows. □

Lemma 10.28. *In a dynamic setting, we can maintain the meta-graph $\mathscr{G}_1 = (\mathscr{Z}_1, \mathscr{E}_1)$ in $O((10/\varepsilon)^{K+4} \cdot n^{1/K})$ update time, amortized over the number of edge insertions/deletions in the input graph $G = (V,E)$.*

Proof. (Sketch) We first describe our algorithm for maintaining the meta-graph $\mathscr{G}_1$.

Whenever an edge (u,v) is inserted into (resp. deleted from) the input graph $G = (V,E)$, we call the procedure described in Section 10.2. This updates the w-structures for levels $\{2, \ldots, K\}$ in a top down manner. When the procedure finishes updating the w-structures for level 2, we consider the set of nodes $L = \bigcup_{j=2}^{K} L_j$. These are only nodes whose capacities in level one need to be updated, for if a node $z \notin L$, then none of its discretized weights $\mathscr{W}_z(w_j)$ were changed, and hence its capacity $b_1^*(z)$ also does not change.

Accordingly, we scan through the nodes in L. For each node $z \in L$, we first update the value of $b_1^*(z)$, and then check all its incident edges $(z,z') \in H_2$ (since $E_1 \subseteq H_2$ by Lemma 9.26). For each such edge $(z,z') \in H_2$, we insert/delete $\kappa_z^* \cdot \kappa_{z'}^* \leq (1/\varepsilon^2)$ meta-edges in $\mathscr{G}_1$, depending on whether the nodes z, z' belong to Z_1 or not. Since $\deg_z(H_2) \leq d_2 = n^{1/K}$ for all nodes z (see Lemma 9.27, Definition 9.8), this procedure takes $O(|L| \cdot (1/\varepsilon)^2 \cdot n^{1/K})$ time. But note that according to our charging scheme in Section 10.3 (see Rule 10.20), each node $z \in L$ has $d_1 \cdot n^{1/K} = n^{1/K}$ dollars deposited into WORK-ACCOUNT$[z,1]$. Thus, we reach the following conclusion:

- The total time spent in maintaining the meta-graph $\mathscr{G}_1$ is at most $(1/\varepsilon)^2$ times the amount of dollars deposited into the work-accounts at level one.

From our framework in Section 10.3, it follows that the amortized update time for maintaining the meta-graph $\mathscr{G}_1$ is at most $(1/\varepsilon^2)$ times the amount of dollars deposited into the bank accounts per edge insertion/deletion in G. Accordingly, we get an amortized update time of $O((1/\varepsilon)^2 \cdot (1/\varepsilon)^{K+2} \cdot n^{1/K}) = O((1/\varepsilon)^{K+4} \cdot n^{1/K})$. □

Corollary 10.29. *The number of edge insertions/deletions in $\mathscr{G}_1$ is at most $O((10/\varepsilon)^{K+4} \cdot n^{1/K})$ times the number of edge insertions/deletions in the input graph G.*

Proof. Let t be the ratio between the number of edge insertions/deletions in $\mathscr{G}_1$ and the number of edge insertions/deletions in G. The corollary follows from Lemma 10.28 and the fact that t cannot be larger than the amortized update time for maintaining $\mathscr{G}_1$. □

Theorem 10.30. *We can maintain the value of w_1^* in $O((10/\varepsilon)^{K+8} \cdot n^{2/K})$ update time, amortized over the number of edge insertions/deletions in the input graph $G = (V,E)$.*

Proof. By Invariant 9.21, the size of w_1^* gives a $(1+\varepsilon)$-approximation to the maximum possible size of a fractional b-matching in $G_1 = (Z_1, E_1)$ with respect to the node-capacities $\{b_1^*(z)\}$. We will maintain a matching $\mathscr{M}_1' \subseteq \mathscr{E}_1$ in the meta-graph $\mathscr{G}_1 = (\mathscr{Z}_1, \mathscr{E}_1)$ whose size will be a $(1+\varepsilon)$-approximation to the size of the maximum matching in $\mathscr{G}_1$. By Theorem 10.27, the quantity $\varepsilon \cdot |\mathscr{M}_1'|$ will serve as an accurate estimate for the size of w_1^*.

Corollary 9.28 and Definition 9.11 guarantee that $\deg_z(E_1) \leq d_1 \cdot n^{1/K} = n^{1/K}$ for all nodes $z \in Z_1$. Since each edge $(u,v) \in E_1$ leads to $\kappa_u^* \cdot \kappa_v^* \leq (1/\varepsilon)^2$ meta-edges, we infer that the maximum degree of a meta-node in $\mathscr{G}_1$ is $d^* = (1/\varepsilon^2) \cdot n^{1/K}$. Hence, using a recent result of Gupta and Peng [7], we can maintain the matching $\mathscr{M}_1'$ in $O(d^*/\varepsilon^2)$ update time, amortized over the number of edge insertions/deletions in $\mathscr{G}_1$. Using Corollary 10.29, we get an update time of $O((d^*/\varepsilon^2) \cdot (10/\varepsilon)^{K+4} \cdot n^{1/K}) = O((10/\varepsilon)^{K+8} \cdot n^{2/K})$, amortized over the number of edge insertions/deletions in the input graph. Finally, note that this subsumes the update time for maintaining the meta-graph $\mathscr{G}_1$, as derived in Lemma 10.28. □

10.5 Maintaining a $(1+\varepsilon)$-approximation to the size of w^r

Consider any level $i \in \{2,\ldots,K\}$. From Observation 9.23 and Definition 9.13, we infer that for every node $v \in T_i \cup S_i$ the residual capacity $b_i^r(v)$ is a positive integral multiple of ε. Further, equation 165 and Definition 9.13 ensures that $b_i^r(v) \leq 1$ for every node $v \in T_i \cup S_i$. Thus, we get:

Observation 10.31. *For every node $v \in T_i \cup S_i$, with $i \in \{2,\ldots,K\}$, we have the residual capacity $b_i^r(v) = \kappa_v^r \cdot \varepsilon$ for some integer $\kappa_v^r \in [1, 1/\varepsilon]$.*

Similar to Section 10.4, we now define a "meta-graph" $\mathscr{G}_i = (\mathscr{Z}_i, \mathscr{E}_i)$ at each level $i \in \{2,\ldots,K\}$. For clarity of exposition, we will refer to the nodes in $\mathscr{Z}_i$ as "meta-nodes" and the edges in $\mathscr{E}_i$ as "meta-edges". The meta-graph is constructed as follows. For each node $v \in T_i \cup S_i$, we create κ_v^r meta-nodes $v(1),\ldots,v(\kappa_v^r)$, where κ_v^r is defined as in Observation 10.31. Next, for each edge $(u,v) \in E_i^r$ (see Invariant 9.14), we create $\kappa_u^r \cdot \kappa_v^r$ meta-edges $\{(u(j), v(l))\}, 1 \leq j \leq \kappa_u^r, 1 \leq l \leq \kappa_v^r$. The next theorem relates the fractional matchings in the meta-graph $\mathscr{G}_i$ with the fractional b-matchings in the residual graph $G_i^r = (T_i \cup S_i, E_i^r)$.

Theorem 10.32. *Consider any level $i \in \{2,\ldots,K\}$.*

1. *Every fractional matching w_i'' in the meta-graph $\mathscr{G}_i$ corresponds to a unique fractional b-matching w_i' in the residual graph G_i^r with respect to the node-capacities $\{b_i^r(z)\}$. The size of w_i'' is exactly $(1/\varepsilon)$-times the size of w_i'.*
2. *A maximal fractional matching in $\mathscr{G}_i$ corresponds to a maximal fractional b-matching in G_i^r with respect to the node-capacities $\{b_i^r(z)\}$.*

Proof. (Sketch) Given a fractional matching w_i'' in $\mathscr{G}_i$, we construct the corresponding fractional b-matching w_i' in G_i^r as follows. Initially, we set $w_i(e) \leftarrow 0$ for every edge $e \in E_i^r$. Next, we scan through the list of meta-edges in $\mathscr{E}_i$. For every meta-edge of the form $(u(j), v(l)), 1 \leq j \leq \kappa_u^r, 1 \leq l \leq \kappa_v^r$, we set $w_i'(u,v) \leftarrow w_i'(u,v) + \varepsilon$. It is easy to check that this construction satisfies both the properties stated in the theorem. □

Lemma 10.33. *In a dynamic setting, we can maintain all the meta-graphs $\{\mathscr{G}_i = (T_i \cup S_i, \mathscr{E}_i)\}, 2 \leq i \leq K$, in $O((10/\varepsilon)^{K+4} \cdot n^{1/K})$ update time, amortized over the number of edge insertions/deletions in $G = (V,E)$.*

Proof. (Sketch) Similar to the proof of Lemma 10.28. □

Corollary 10.34. *The number of edge insertions/deletions in $\bigcup_{i=2}^{K} \mathscr{E}_i$ is at most $O((10/\varepsilon)^{K+4} \cdot n^{1/K})$ times the number of edge insertions/deletions in G.*

Proof. (Sketch) Similar to the proof of Corollary 10.29. □

Theorem 10.35. *There is a dynamic algorithm with the following properties:*

1. *It maintains a $(1+\varepsilon)^2$-approximation to the size of w^r.*
2. *Its update time is $O((10/\varepsilon)^{K+6} \cdot n^{1/K} \cdot \log n)$, amortized over the edge insertions/deletions in G.*

Proof. (Sketch) If we could maintain a maximal fractional matching in $\mathscr{G}_i$ for all $i \in \{2, \ldots, K\}$, then we would have had an exact estimate of the size of w_i^r for all $i \in \{2, \ldots, K\}$ (see Theorem 10.32, Invariant 9.14). Instead, we use the dynamic algorithm of Bhattacharya, Henzinger and Italiano [5] to maintain a $(1+\varepsilon)^2$-maximal fractional matching in each $\mathscr{G}_i$.[8] This means that we get a $(1+\varepsilon)^2$-approximation to the size of each w_i^r, and hence a $(1+\varepsilon)^2$-approximation to the size of $w^r = \sum_{i=2}^{K} w_i^r$.

Using the analysis of Bhattacharya, Henzinger and Italiano [5], the update time of this dynamic algorithm is $O(\log n/\varepsilon^2)$, amortized over the number of edge insertions/deletions in $\bigcup_{i=2}^{K} \mathscr{E}_i$. Applying Corollary 10.34, we get a update time of $O((10/\varepsilon)^{K+6} \cdot n^{1/K} \cdot \log n)$, amortized over the number of edge insertions/deletions in G. Note that this subsumes the update time for maintaining the meta-graphs $\{\mathscr{G}_i\}, 2 \leq i \leq K$, as derived in Lemma 10.33. □

[8] In such a fractional matching w_i'', for every meta-edge $(z, z') \in \mathscr{E}_i$, either $W_z(w_i'') \geq 1/(1+\varepsilon)^2$ or $W_{z'}(w_i'') \geq 1/(1+\varepsilon)^2$.