Approximation Algorithms for Stochastic Matchings and Independent Sets

By

Joydeep Mukherjee

MATH10201004008

The Institute of Mathematical Sciences, Chennai

A thesis submitted to the

Board of Studies in Physical Sciences

In partial fulfillment of requirements

For the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

February, 2019

Homi Bhabha National Institute

Recommendations of the Viva Voce Board

As members of the Viva Voce Board, we certify that we have read the dissertation prepared by **Joydeep Mukherjee** entitled "Approximation Algorithms for Stochastic Matchings and Independent Sets" and recommend that it maybe accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

V. Arvind Date: 1-2-2019

Chair - V Arvind

Date: 01/02/2019

Guide/Convener - C R Subramanian

N.S.N Date: 1/2/19

Examiner - N S Narayanaswamy

Sharma Date: 01/02/2019

Member 1 - Vikram Sharma

Mmahajan Date: 1 Feb 2019

Member 2 - Meena Mahajan

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to HBNI.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it may be accepted as fulfilling the dissertation requirement.

Date: Feb 01, 2019

Place: Chennai

Guide: C R Subramanian

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at Homi Bhabha National Institute (HBNI) and is deposited in the Library to be made available to borrowers under rules of the HBNI.

Brief quotations from this dissertation are allowable without special permisiion, provided that accurate acknowledgement of source is made. Requests for permisiion for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the Competent Authority of HBNI when in his or her judgement the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

Joydeep Mukherjee

DECLARATION

I, hereby declare that the investigtion presented in the thesis has been carried out by me. The work is original and has not been submitted earlier as a whole or in part for a degree / diploma at this or any other Institution / University.

Joydeep Mukherjee

LIST OF PUBLICATIONS ARISING FROM THE THESIS

Journal

1. Joydeep Mukherjee and C. R. Subramanian. Greedy Heuristics and Stochastic Matchings. In *Asian Journal of Mathematics and Applications*, Vol 2018.

Conferences

1. Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. Improved approximation algorithms for stochastic matching. In *Algorithms-ESA 2015*, pages 1 – 12. Springer, 2015.

2. Abhiruk Lahiri, Joydeep Mukherjee, and CR Subramanian. Maximum independent set on B1-VPG graphs. In *Combinatorial Optimization and Applications*, pages 633– 646. Springer, 2015.

Others

1. Joydeep Mukherjee and C. R. Subramanian. Approximating MIS on B_1, B_2-VPG Graphs, Manuscript, September 2016, Submitted to a journal

2. Abhiruk Lahiri, Joydeep Mukherjee and C. R. Subramanian. On approximating MIS over B_1-VPG graphs, Manuscript, September 2016, Submitted to a journal

Joydeep Mukherjee

DEDICATIONS

Dedicated to my parents.

ACKNOWLEDGEMENTS

Every work is a team effort. And this is the space where I can express my gratitude to at least a part of the team that helped me in making this thesis realized. There are people around me who have made all possible efforts (may be non-academically) to see this day. My deepest regards and heartfelt thanks to all of them and I beg your pardon for the fact that I am not including your names explicitly here. I keep my list limited to exactly to those people from academia who directly helped me realizing this thesis. Still the list is no way complete.

I would begin by extending my deepest gratitude to my advisor Prof. C. R. Subramanian who had made tireless efforts in making this thesis see the light of day. He would always provide me with his valuable time whenever I needed it for any technical and non-technical discussion. These discussions often helped me make my ideas more precise and which would eventually lead us to answer many questions some of which are presented in this thesis. I also thank him for investing considerable time in designing and working out the detailed proofs of some of the algorithmic results. My heartfelt thanks to you!

I would also thank all the TCS faculty members at IMSc: Professors Kamal Lodaya, R. Ramanujam, V. Arvind, Meena Mahajan, Venkatesh Raman, C. R. Subramanian, Saket Saurabh and Vikram Sharma who taught me fundmentals of Theoretical Computer Science with utmost care. Thank you all for showing me the treasure trove called Theoretical Computer Science.

I would also take this opportunity to thank Prof. Saket Saurabh who had helped me in several ways during my stay at MatScience.

I also thank all staff members of MatScience who helped me in umpteen number of ways during my stay here.

I render my thanks to all my friends in MatScience: Sheeraz, Ramachandra, Raja, Gaurav, Anup, Shukla, Shankardeep, Niranka, Kunal, Fahad, Atanu, Shibasish, Arghya, Nitesh, Abhiruk, Anuj, Swaroop, Devanand, Chandan, Prateep. They would always stand by me in any sort of difficult situation. Thank you friends!

I would specially thank Prof. Sandip Das from ISI, Kolkata who rendered every possible help during my Ph.D. period. Thank you Sir!

I also thank my friends at ISI and Ramakrishna Mission Vivekananda University: Arijit (SMU), Arun, Dibyayan, Harminder, Subhadeeep, Sagnik, Sanjana, Soumen da, Tanmoy (Vivekananda University), Umakant.

Thank you all around me whose name I did not include here but whose help was no way less. I convey my deepest and heartfelt regards and thanks to all of them.

Contents

Abstract

We study the weighted version of the stochastic matching (under the *probe-and-commit* model) as introduced by Chen et al. [CIK+09]. As input a random subgraph H of a given edge-weighted graph $(G = (V, E), \{w_e\}_e)$ (where each edge $e \in E$ is present in H independently with probability p_e) is revealed (on a *probe-and-commit* basis). Our goal is to design an efficient adaptive algorithm that builds a matching by probing selectively edges of E for their presence in H subject to obeying the following two constraints on probing : (*i*) include an edge irrevocably in the matching if it is found to exist after it is probed, (*ii*) the number of probes involving a vertex v cannot exceed a nonnegative parameter t_v known as v's patience. All of G, $\{w_e\}_e$, $\{p_e\}_e$ and $\{t_u\}_u$ is revealed to the algorithm before its execution. The performance of the algorithm is measured by the expected weight of the matching it produces. For approximation measures, it is compared with the expected weight of an optimal adaptive algorithm for the input instance.

We analyze a natural greedy algorithm for this problem and obtain an upper bound of $\frac{2}{p_{min}^2}$ on the approximation factor of its performance. Here, p_{min} refers to $\min_{e \in E} p_e$. No previous analysis of any greedy algorithm for the weighted stochastic matching (under the probe-and-commit model) is known. We also analyze another greedy heuristic and establish that its approximation ratio can become arbitrarily large even if we restrict ourselves to unweighted instances.

Bansal et al., [BGL+10] introduced an online variant of weighted bipartite stochastic matching. They presented an LP-based algorithm with an approximation ratio of 7.92.

We present a new algorithm (also LP-based) for the same problem which improves the approximation ratio to 5.2.

We present approximation algorithms for the maximum independent set (MIS) problem over the class of B_1, B_2-VPG graphs and also for the subclass, equilateral B_1-VPG graphs. We first show an approximation guarantee of $O((\log n)^2)$ for the MIS problem of B_1-VPG graphs. Then we improve the approximation factor to $O(\log n)$ for the MIS problem of B_1-VPG graphs. For the equilateral B_1-VPG graphs we show an approximation guarantee of $O(\log d)$ where d denotes the ratio d_{max}/d_{min} and d_{max} and d_{min} denote respectively the maximum and minimum length of of any arm in the input B_1-VPG representation of the graph. The NP-completeness of the decision version restricted to unit length equilateral B_1-VPG graphs is also established. For B_2-VPG graphs we present an approximation algorithm whose approximation guarantee is $O((\log n)^2)$.

List of Figures

Chapter 1

Introduction

Combinatorial optimization problems are ubiquitous in today's society. But most of the interesting combinatorial optimization problems are NP-hard. There are various ways to deal with such hard problems. One way is to obtain an optimal solution by employing an algorithm which is likely to require an exponential amount of time. Such algorithms fall under the purview of exact algorithms. Another way is instead of designing an efficient algorithm which produces a solution which is not optimal, we design an algorithm which gives a solution that is not far from an optimal solution in terms of quality. This is roughly the purview of approximation algorithms. In this thesis, we deal with such a type of algorithms. Below, we present the following formal definition of an approximation algorithm as provided in [WS11].

Definition 1. *An $\alpha(n)$-approximation algorithm for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a multiplicative factor of $\alpha(n)$ of the value of an optimal solution. n stands for the size of the input.*

The above definition captures the scenario in which the input is an arbitrary but a deterministic one. For the stochastic version, we provide a definition while presenting the approximation algorithms.

In this thesis, we present approximation algorithms for two combinatorial optimization problems. They are *stochastic matching* and *maximum independent set(MIS)* for B_k-VPG graphs.

1.1 Stochastic Matching:

Matching in a graph is a set of edges such that no two edges share a common vertex. The matching problem is to produce, given a nonnegatively edge-weighted graph, a matching of maximum total weight. This problem is well-known to be solvable in polynomial time. For bipartite graphs, there are several polynomial time algorithms like Ford-Fulkerson algorithm [FF56], Hopcroft-Karp algorithm [HK73] to name a few. For general graphs, the Edmond's algorithm [Edm65] solves the problem in polynomial time.

In the first part of the thesis, we study the stochastic version of the matching problem. We study the stochastic matching problem in both offline and online settings. We introduce them one by one below.

Offline Setting: As input, a random subgraph H of a graph $G = (V, E)$ (where each edge $e \in E$ is present in H independently with probability p_e) is revealed (on a *probe-and-commit* basis) along with (*i*) a positive weight w_e for every $e \in E$ and also (*ii*) a nonnegative integer t_v (for each $v \in V$) called the *patience parameter* of v. Our goal is to design an efficient adaptive algorithm that builds a matching by probing selectively edges of E for their presence in H subject to the following two constraints on probing: (*i*) include an edge irrevocably in the matching if it is found to exist after it is probed, (*ii*) the number of probes involving a vertex cannot exceed its patience. The performance of the algorithm is measured by the expected weight of the matching it produces. For approximation measures, it is compared with the expected weight of an optimal adaptive algorithm for the input instance.

The work on approximation algorithms for the offline unweighted stochastic matching

was initiated by Chen et al. [CIK+09]. Later on, Adamczyk [Ada11] proved that a greedy algorithm considered in [CIK+09] is indeed a 2-approximate algorithm. Subsequently, approximate algorithms were obtained by Bansal et al. [BGL+10] for the weighted case. Gupta and Nagarajan [GN13] study a generic notion of stochastic probing problems which also specializes to the stochastic matching problem. Adamczyk et al. [ASW13] extend this work to also include submodular functions. The same problem is also studied for special classes of graphs by Molinaro et al. [MR11], both theoretically and experimentally. A sampling-based algorithm was proposed by Costello et al. [CTT12] for the offline weighted version with unbounded patience parameters for vertices.

Online Setting: For the online version, we focus on bipartite graphs. Bansal et al. [BGL+10] introduced this online version. In this version, the algorithm has constraints on the choice and order of edges to be probed. In particular, there is a linear ordering on the vertices (say, the arrival order) of one partite set and the algorithm has to make a decision (on whether to probe or not) for every edge (if any) incident on a just arrived vertex before it considers edges incident on future vertices. It models the sale of items from a set A to buyers arriving in an online fashion. Each buyer has to be processed before we consider the next arriving buyer. The processing of each buyer involves showing a select subset of items in some order until the buyer likes an item (if it happens) in which case both the item and the buyer are removed from the picture. To each buyer, we can associate a type/profile and the type characterizes (*i*) the patience t_b, (*ii*) probability p_{ab} that a buyer of type b buys item a, and (*iii*) w_{ab} the revenue generated if it happens. The type of each arriving buyer is independently and identically distributed over the set B of types. Here, the buyers arrive online. The number of buyers that are going to arrive is known to the algorithm. The goal is to design an efficient online algorithm which produces a matching whose expected revenue is as large as possible. The performance of the algorithm is compared with the expected revenue from the matching produced by an optimal strategy.

The study of the online stochastic matching problem started with the work of Feldman

et al., [FMMM09] and led to further works like those of Bansal et al., [BGL+10]. Some recent improvements have been obtained for the stochastic setting (without the probe-and-commit and tolerance requirements) [BK10, MGS12]. The problem of online stochastic matching considered here differs in the following aspect that the buyers can only see a limited number of items and the buyer buys the first item it likes.

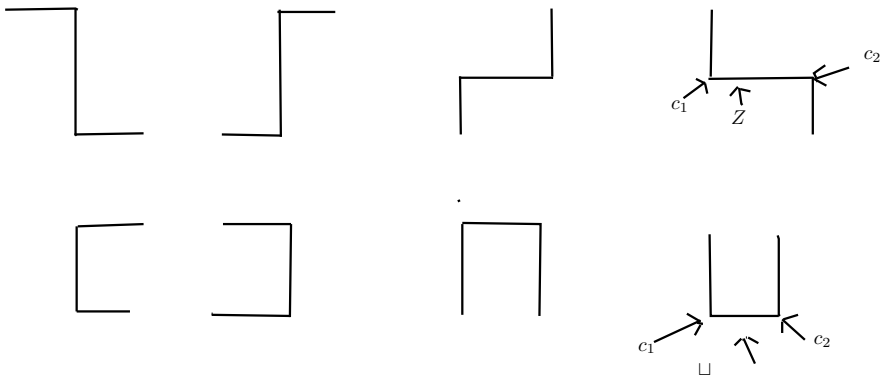
1.2 MIS for VPG-graphs

The problem of computing a maximum independent set (*MIS*) in an arbitrary graph is notoriously hard, even if we aim only for a good approximation to an optimum solution. It is known that, for every fixed $\epsilon > 0$, MIS cannot be approximated within a multiplicative factor of $n^{1-\epsilon}$ for a general graph, unless $NP = ZPP$ [Hås96]. Throughout, n stands for the number of vertices in the input graph. Naturally, there have been algorithmic studies of this problem on special classes of graphs like : (*i*) efficient and exact algorithms for perfect graphs, (*ii*) linear time exact algorithms for chordal graphs and interval graphs, (*iii*) $O(n^2)$ time exact algorithms for comparability and co-comparability graphs, (*iv*) PTAS's (polynomial time approximation schemes) for planar graphs [Bak94] and unit disk graphs [HMR+98], and (*v*) efficient $(\frac{k}{2} + \epsilon)$-approximation algorithms for $(k+1)$claw-free graphs [Hal95].

Geometric objects of various shapes form interesting classes of intersection graphs. These are graph classes for which several algorithmic studies have been carried out for the MIS problem. Approximation algorithms with good approximation guarantees have been obtained. One such class of geometric intersection graphs are B_k-VPG graphs, for $k \geqslant 1$. In this thesis, we focus on B_1-VPG and B_2-VPG classes. Before describing these classes, we provide a brief introduction to the class of VPG graphs.

Vertex intersection graphs of Paths on Grid (or, in short, VPG graphs) was first introduced by Golumbic et al. [ACG+12]. For a member of this class of graphs, its vertices represent

paths joining grid-points on a rectangular planar grid which are a combination of alternate vertical and horizontal segments and two such vertices are adjacent if and only if the corresponding paths intersect. If paths on the grid have at most k bends (90° turns), then the graph is called a B_k-VPG graph. Thus, B_1-VPG (B_2-VPG) graphs denote the class of intersection graphs of paths on a grid where each path has at most one (two) bends. Without loss of generality, B_1-VPG graphs are intersection of the following shapes ⌞, ⌜, ⌝ and ⌟. B_2-VPG graphs are intersection graphs of the following shapes shown in the figure along with the shapes in B_1-VPG graphs.



There are 8 possible shapes for B_2-VPG. We have labelled what we consider the Z-shape and ⊔ shape.

Figure 1.1: There are 8 possible shapes for B_2-VPG graphs with exactly 2 bends. We have labelled what we consider the Z-shape and U shape.

We often refer to a B_1-VPG graph, representable with only paths of type ⌞ as a L-graph. We call a L-shape *equilateral* if the horizontal and the vertical arms are of the same length. The graph which is an intersection graph of equilateral L's is called an *equilateral L-graph*.

The study of MIS for B_k-VPG graphs is motivated from both an algorithmic point of view as well as an application point of view. We first mention our motivations which come from a theoretical point of view. It has been shown that every planar graph has a B_2-VPG representation [CU12] and that every triangle-free planar graph has a B_1-VPG representation [BD15]. In the case of planar graphs, it is already known that the decision version of MIS is NP-complete [GJ77] and also that MIS admits a PTAS [Bak94]. This

naturally motivates us to study the complexity of approximating MIS over B_1-VPG graphs and B_2-VPG graphs because they are respectively the superclasses of triangle-free and general planar graphs. Currently, the only known approximation results for MIS over these classes of graphs are those for the larger and more general class of string graphs, the best of which is an algorithm with an approximation factor of n^{ϵ} (for some fixed $\epsilon > 0$) [FP11]. String graphs are intersection graphs of simple, continuous curves in the plane [ACG+12]. In this context, it would be interesting to know if MIS can be approximated in a better way when restricted to subclasses like B_1, B_2-VPG graphs. The practical motivation for B_k-VPG graphs is from VLSI circuit design. Since wires in a VLSI circuit correspond to paths on a grid and since intersection between wires is to be minimized, the MIS problem represents a finding a large collection of mutually non-intersecting paths on the grid.

Assumption : Without loss of generality (in the context of approximating MIS) and for the ease of describing the arguments, we assume that B_k-VPG graphs are intersection graphs of paths with *exactly* k bends.

Relationships between other known graph classes and VPG graphs have been studied in [ACG+12]. In [CU13], it has been shown that planar graphs form a subset of B_2-VPG graphs. Recently, this result has been further tightened by Therese Biedl and Martin Derka. They have shown that planar graphs form a subset of 1-string B_2-VPG graphs [BD15] which is a subclass of B_2-VPG graphs. In [FKMU14], authors have shown that any full subdivision of any planar graph is a L-graph. By a full subdivision of a graph G, we mean a graph H obtained by replacing every edge of G by a path of length two or more with every newly added vertex being part of exactly one path. They have also shown that every co-planar graph (complement of a planar graph) is a B_{19}-VPG graph. A relationship between poset dimension and VPG bend-number has also been obtained in [CGTW15]. Contact representation of L-graphs has been studied in [CKU13]. In this work, the authors have studied the problems of characterizing and recognizing contact L-graphs and have

also shown that every contact L-representation has an equivalent equilateral contact L-representation. By a *contact* L-representation, we mean a more restricted intersection, namely, that two vertices are adjacent if and only if the corresponding Ls just touch. Recognizing VPG graphs is shown to be NP-complete in [CJKV12]. In the same work, it is also shown that recognizing if a given B_{k+1}-VPG graph is a B_k-VPG graph is NP-complete even if we are given a B_{k+1}-VPG representation of the input. The recognition problem has also been looked at for some subclasses of B_0-VPG graphs in [CCS11].

Apart from works mentioned in the previous sections, there has been quite a lot of work on independent sets of geometric intersection graphs. For intersection of axis-parallel rectangles, an $O(\log n)$-approximation algorithm for MIS is presented in [AVKS98, BDMR01, KMP98, Nie00]. There is an $O(\log \log n)$-approximation algorithm for MIS of unweighted axis parallel rectangles obtained by Chalermsook and Chuzoy in [CC09] which is still the best known for the unweighted case. For the weighted case, the best known one was obtained by Chan and Har-Peled in [CHP12] which has an approximation factor of $O(\frac{\log n}{\log \log n})$. QPTAS for MIS problems in axis-parallel rectangles was first proposed by Adamaszek and Wiese in [AW13]. This was later generalized to QPTAS for MIS for the case of general polygons in [AW14, HP14]. For segment graphs, there is an $O(n^{\frac{1}{2}})$-approximation algorithm by Agarwal and Mustafa [AM04].

1.3 Thesis Outline

Chapter 2 deals with the analysis of the greedy (based on choosing the edge with maximum expected contribution) algorithm for the weighted stochastic matching problem. In this chapter, we analyze the algorithm for its performance guarantee and obtain both an upper bound as well as a lower bound on its worst-case value. We establish that its approximation ratio is at most $\frac{2}{p_{min}^2}$, where $p_{min} = \min\{p_e : e \in E\}$. We also exhibit and analyze an explicit and infinite family of weighted graphs where the approximation ratio

can become as large as $\frac{2}{p_{min}}$. Since this variant selects edges for probing based on their individual expected contribution, it can be thought of as being greedy edge-wise. We also propose a simple variant of the greedy approach which can be thought of as being greedy vertex-wise and also a generalization of both greedy algorithms (vertex-wise as well as edge-wise).

In Chapter 3, we provide an analysis of online stochastic matching. We propose and analyze a new LP-based algorithm and establish that it is 5.2-approximate. We adopt an LP (Linear Programming) based approach along with dependent randomized rounding to obtain the approximation guarantee.

In Chapter 4, we present an $O((\log_2 n)^2)$-approximation algorithm for the MIS problem over B_1-VPG graphs. In the same chapter, we present an $O(\log_2 2d)$-approximation for equilateral L-graphs where d denotes the ratio between the lengths of longest and shortest horizontal arms of members of the given equilateral L-graph. If the lengths of the equilateral L's are all equal to 1 unit, then we call the corresponding intersection graph an unit L-graph. We also establish that the decision version of the MIS problem over unit L-graphs is NP-complete. For the design of approximation algorithms, we use some combinatorial observations and the divide and conquer approach to obtain the approximation guarantees mentioned before.

In Chapter 5, we present an algorithm with an improved approximation ratio of $O(\log_2 n)$ for the MIS problem over B_1-VPG graphs. This improvement is achieved by devising an exact algorithm (for a special subclass of B_1-VPG graphs) and combine it with a divide and conquer approach.

Chapter 6 presents new approximation algorithms for the MIS problem over B_2-VPG graphs and an upper bound of $O(\log_2 n)^2$ is established on its approximation ratio. This improves the bound of n^ϵ (for some $\epsilon > 0$) on the ratio of the previously best algorithm. Our main ingredient in obtaining this improvement is again an exact algorithm for a special subclass of B_2-VPG graphs combined with an application of the divide and conquer

paradigm.

Finally, in Chapter 7, we conclude with some open problems.

Chapter 2

Greedy Analysis of Stochastic Matching

2.1 Introduction

The Greedy heuristic being one of the simplest algorithmic approaches has a unique place in combinatorial optimization. It is always worth looking at its performance and gather to know its power and limitations. In particular, the performance of the Greedy algorithm for computing a large matching under different settings has been studied both for arbitrary graphs (for its worst case perfomance) (see [KH78], [GS62]) and as well as for random instances (for its average case performance) (see [DF91], [DFP93], [AFP98], [FRS93]). In this chapter, we study the performance of the greedy heuristics on the weighted stochastic matching problem, a natural stochastic variant of the maximum matching problem.

A typical input instance of this problem is a 4-tuple $(G = (V, E), \{t_u\}_{u \in V}, \{p_e\}_{e \in E}, \{w_e\}_{e \in E})$ where $G = (V, E)$ is an weighted graph, each t_u is a nonnegative integer (known as the patience of u). Consider a random spanning subgraph H where each $e \in E$ is present in H independently with probability p_e and where H is revealed on a probe-and-find basis. Our goal is to design an efficient algorithm (possibly adaptive, possibly randomized) to find a matching in H and which works by probing selectively edges of E for their presence in H subject to the following two constraints on probing : (*i*) *commitment*: include an edge ir-

revocably in the matching if it is found to exist after it is probed, (*ii*) *patience*: the number of probes involving a vertex cannot exceed its patience. The performance of the algorithm is measured by the expected total weight of the matching it produces. For approximation measures, it is compared with the expected weight of an optimal adaptive algorithm for the input instance. An optimal strategy is one for which the expected weight of the solution it produces its maximum over all adaptive strategies. We use interchangeably the terms adaptive algorithm and strategy. Note that all edges of G need not be probed and hence all edges of H may not be discovered by the algorithm.

The *unweighted* stochastic matching problem (with probing commitments) models some practical optimization problems like maximizing the expected number of kidney transplants in the kidney exchange program (see [CIK+09] for details). This problem was introduced by Chen et al. [CIK+09] and they analyzed a greedy algorithm to solve it and proved that the greedy algorithm produces a solution of expected size at least a quarter of the expected size of an optimal strategy. This gives us a 4-approximate algorithm. It was also conjectured that their greedy algorithm is a 2-approximate algorithm. This was later affirmatively verified by Adamczyk [Ada11].

In this work, we study the offline, weighted version of the stochastic matching problem. In the offline version, the algorithm, after processing the entire input information ($G = (V, E)$, $\{t_u\}_u$, $\{w_e\}_e$ and $\{p_e\}_e$) that is revealed before-hand, can choose any adaptive strategy to probe the edges.

We analyze several variants of the greedy approach to solve this problem. In Section 2.3, we propose and analyze a natural greedy variant which always probes an edge with the highest expected weight it contributes (if probed) and establish that its approximation ratio is at most $\frac{2}{p_{min}^2}$, where $p_{min} = \min\{p_e : e \in E\}$. This affirmatively confirms a claim presented in [CIK+09] (without details) that the approximation factor of the greedy algorithm for the weighted version can be unbounded. It also follows that approximation ratio is less than 4 on general weighted graphs if $p_{min} > \frac{1}{\sqrt{2}}$. Since this variant selects edges

for probing based on their individual expected contribution, it can be thought of as being greedy edge-wise and denote it by **GRD-EW** . Our result is the first analysis of a greedy heuristic for stochastic matching on weighted graphs. The precise statement of our result is as follows.

Theorem 1. **GRD-EW** *is a* $\frac{2}{p_{min}^2}$*-approximate algorithm for the weighted stochastic matching problem.*

We also show that the inverse dependence on p_{min} cannot be completely eliminated by a more careful analysis even if we allow every vertex to probe all edges incident at it (that is $t_u \geqslant d_u$ for every u). Thus, we obtain a lower bound on the worst-case approximation ratio of **GRD-EW** for the weighted case. This is stated in the following lemma whose proof is provided in Section 2.3.

Lemma 1. *There exists an infinite and explicit family* $\{(G_n, t_n)\}_n$ *of weighted input instances (with unlimited patience values) such that the expected weight of the solution produced by* **GRD-EW** *is smaller than the expected weight of an optimal strategy by a multiplicative factor of nearly* $\frac{2}{p_{min}}$.

Since the algorithm works by probing edges, we model the execution of an algorithm as a full binary decision tree as in [CIK+09, Ada11]. Adamcyzk [Ada11] presents a very careful analysis of the decision tree to prove that the greedy algorithm is a 2-approximate algorithm for the unweighted version. Our analysis is inspired by the analysis of [Ada11] and we borrow some of the notions and notations from this work. However, ours is not a straighforward generalization to the weighted version and some non-trivial issues (arising for the more general weighted case) have to be handled while analyzing the greedy heuristic.

In Section 2.4, we propose a simple variant **GRD-VW** of the greedy approach which can be thought of as being greedy vertex-wise. Here we define a notion of revenue m_u associ-

ated with a vertex u. For a given set S of l edges incident at a vertex u, an *optimal ordering* of S is any linear ordering σ over S such that if members of S are probed consecutively as per σ, then the expected contribution $E_{S,\sigma}$ from these probings maximized. It can be verified that an optimal ordering is any ordering obtained by sorting the edges in decreasing order of their weights. For a vertex u, let m_u denote the expected contribution one obtains by probing edges of S_u in an optimal order. Here, S_u is the set of t_u edges incident at u having the k largest expected contributions $w_e p_e$. The **GRD-VW** proceeds by choosing that vertex u for which the revenue m_u is maximized and then probes edges in S_u in an optimal order and decreases the tolerances appropriately after each probe. We prove that the worst-case approximation ratio of **GRD-VW** can be unbounded even if we restrict ourselves to the unweigted instances (the case for which **GRD-EW** is a 2-approximation algorithm). Formally stated, we have the following result which is proved in Section 2.4.

Lemma 2. *There exists an infinite and explicit family* $\{(G_n, t_n)\}_{n\geqslant 1}$ *of unweighted input instances such that the expected size of the solution obtained by* **GRD-VW** (G_n, t_n) *is smaller than that of an optimal strategy by a multiplicative factor of nearly* $\Omega\left(\frac{1}{p_{max}}\right)$ *where* $p_{max} = \max_e p_e$.

The edge-wise and vertex-wise greedy heuristics **GRD-EW** and **GRD-VW** analyzed in Sections 2.3 and 2.4 can both be thought of as special cases of a more generalized notion of a greedy heuristic. Fix any function $k : \mathcal{N} \to \mathcal{N}$ satisfying $k(n) \leqslant n$ for every n. We define a variant for every fixed choice of k and denote the variant by $GRD_k(G, w, p, t)$ or shortly $GRD_k(G, t)$ if w and p are clear from the context. $GRD_k(,)$ is exactly the same as the vertex-wise variant **GRD-VW** but differs only in the definition of m_v, more precisely, in that m_v is the expected contribution one obtains by probing consecutively $\min\{k(|V|), t_v\}$ heaviest available edges incident at v, with the edges being probed in decreasing order of their weights. When $k(n) = n$ for every n, we obtain that $GRD_k()$ is the same as **GRD-VW** . When $k(n) = 1$ for every n, we obtain that $GRD_k()$ is the same as **GRD-EW** described in Section 2.3. The following lemma establishes that $GRD_k()$ also has unbounded worst-case

approximation ratio for any fixed $k = k(n)$ such that $k \to \infty$ as $n \to \infty$ even if restricted to unweighted instances. The proof is presented in Section 2.5.

Lemma 3. *For any $k = k(n)$ such that (i) $k \leqslant n$, (ii) k divides n and (iii) $k \to \infty$ and for every sufficiently small $\epsilon > 0$, there exists an infinite and explicit family $\{(G_n, t_n)\}_{n \geqslant 1}$ of unweighted input instances such that the expected size of the solution obtained by $GRD_k(G_n, t_n)$ is smaller than that of an optimal strategy by a multiplicative factor which is nearly $\Theta(k^{1-\epsilon})$.*

2.2 Preliminaries

Below, we present some conventions, assumptions and models we will be employing for the rest of this work. Throughout, we consider an instance $I = (G, w, p, t)$ where $G = (V, E)$ is an undirected graph, $w : E \to \mathcal{R}^+$ is the weight function, $t : V \to \mathcal{N}$ is the patience function and $p : E \to [0, 1]$ is the edge probability function. For the sake of simplicity, we often denote this collective input by the short notation (G, t) if the additional inputs $\{p_e\}_e, \{w_e\}_e$ can be inferred from the context.

2.2.1 Convention : rationalization of patience values

We assume, without loss of generality, that $t_u \leqslant d_u$ for every $u \in V$, where d_u is the degree of u in G. Higher values of t_u are not going to lead to better solutions. Throughout the chapter, we always enforce this assumption (wherever it becomes necessary), by invoking a subroutine *Rationalize*(G, t) which, for any vertex u with $t_u > d_u$, redefines t_u to be d_u. Enforcing this assumption helps us to simplify the description of some greedy variants we will study in Sections 2.4 and 2.5.

Also, at any point, the current graph contains only those edges joining vertices with positive patience values. This can be ensured by removing edges incident at vertices whose

patience has been exhausted.

2.2.2 Assumption : normalization of weights

Since multiplying each edge weight by a common factor c does not really change the outcome (except multiplying its total weight by c) of any algorithm, we can normalize all weights by replacing each w_e by $\frac{w_e}{w_{max}}$, where $w_{max} = \max_e w_e$. This normalization simplifies some of the expressions arising in the analysis. In view of this, from now on, *we assume without loss of generality that* $w_e \leqslant 1$ *for each e.*

2.2.3 Modeling algorithms by decision trees

Our focus is on algorithms (possibly adaptive, possibly randomized) which are based on probing edges (with a commitment to inclusion) and we analyze such algorithms using the decision tree model employed in the works [CIK+09, Ada11]. The model is described as follows. Any algorithm *ALG* can be represented by a (possibly) exponential sized full binary tree also denoted by *ALG*. Each internal node represents either probing an edge or tossing a (biased) coin. The coin tosses capture the randomness (possibly) employed by the algorithm. For deterministic algorithms, each internal node will correspond to only an edge probe. An internal node x probing an edge e will be labeled with e and $w_x = w_e$. An internal node x tossing a coin will be labeled by an empty string and $w_x = 0$. Consider an internal node x. If x involves probing an edge e and if the probe is successful, then the algorithm will proceed further as per the strategy specified by the left subtree of x and if it is unsuccessful, it will proceed as per the right subtree. Similarly, if x corresponds to a coin toss, then the algorithm will proceed further as per the strategy specified by the left (or the right) subtree of x depending on whether the toss is successful or not. However, only internal nodes probing edges can make a positive contribution to the weight of the solution found.

We give a recursive definition of a decision tree: The decision tree ALG corresponding to an algorithm ALG on an instance $I = (G, t)$ (ignoring the specification of w_e's and p_e's which are not going to change through the execution) is a rooted full binary tree T (with root r) where

1. r is labelled by the emptyset if G is an empty graph having no edges.

2. r probes an edge $e = \alpha\beta$ if G has at least one edge or r tosses a coin with bias p_r.

3. left edge out of r is labelled by $p_{\alpha\beta}$ if r probes $\alpha\beta$ or is labelled by p_r if r tosses a coin.

4. right edge coming out of r is labelled by $1 - p_{\alpha\beta}$ or by $1 - p_r$ depending on the case.

5. the left subtree of r represents further execution of ALG on on the instance $I_L = (G \setminus \{\alpha, \beta\}, t)$ if r probes $\alpha\beta$. Otherwise, it represents further execution of ALG on I.

6. the right subtree of r represents further execution of ALG on the instance $I_R = (G \setminus \{\alpha\beta\}, t')$ where $t'_\alpha = t_\alpha - 1$, $t'_\beta = t_\beta - 1$ and $t'_\gamma = t_\gamma$ for all other vertices γ if r probes $\alpha\beta$. Otherwise, it represents further execution of ALG on I.

Without loss of generality, we assume that the root r of an optimal tree OPT always probes an edge.

We make use of the following notations. For any algorithm ALG and any node x in ALG, let q_x denote the probability of reaching x in an execution of $ALG(G, t)$. Also, for a node x representing an edge e, we use w_x to denote the weight w_e. It can be verified that the performance of ALG on (G, t) can be expressed as $\mathbb{E}[ALG] = \sum_{x \in ALG} q_x p_x w_x$ where the summation is over all internal nodes.

2.3 Greedy heuristic for the weighted version

We focus on the offline version. This means that the input I consisting of the random model $(G = (V, E), \{p_e\}_{e \in E})$ alongwith the additional inputs $(\{w_e\}_{e \in E}, t = \{t_u\}_{u \in V})$ will be revealed to the algorithm before its execution. After some preprocessing, the algorithm can choose to select and probe the edges in any order of its choice. We analyze the following greedy algorithm for the above problem. We use Gr to denote both the greedy algorithm and the corresponding decision tree. Let $\alpha\beta$ be the first edge probed by $Gr(G, t)$. This means that $w_{\alpha\beta}p_{\alpha\beta}$ maximizes $w_e p_e$ over all edges e. We also use OPT to denote any optimal strategy for I and also the associated decision tree. It also denotes the weight of the matching produced by OPT when executed on I.

Algorithm 1 Greedy Algorithm $Gr(G, t)$:

1: $E' \leftarrow E$. $M \leftarrow \emptyset$.
2: **while** $E' \neq \emptyset$ **do**
3: Choose an arbitrary edge $e = uv \in E'$ which maximizes $w_e p_e$.
4: Probe e and add e to M if e is found to be present.
5: **If** $e \in M$, then set each of t_u and t_v to be zero; **else** decrement t_u and t_v.
6: Remove e from E'.
7: Remove any edge in E' incident at u (v) if t_u (t_v) equals zero.
8: Rationalize(G, t).
9: **endwhile**
10: Output M.

To analyze the performance of $Gr(G, t)$, we study the following two algorithms ALG_L and ALG_R introduced and defined as in [Ada11] to work on instances I_L and I_R respectively. By an $\alpha\beta$-probe (α-probe or β-probe) of $OPT(G, t)$, we mean probing edge $\alpha\beta$ (probing edge $\alpha\gamma$ for some $\gamma \neq \beta$ or probing edge $\delta\gamma$ for some $\delta \neq \alpha$).

The algorithm ALG_L mimics the execution of $OPT(G, t)$ except that it replaces each $\alpha\beta$-probe, each α-probe and each β-probe by an appropriate coin toss. That is, whenever there is such a probe (at a node x of $OPT(G, t)$) of an edge e incident at either α or β or both, a coin with bias p_e is tossed. With probability p_e, ALG_L mimics the left subtree of x and with probability $1 - p_e$ it mimics the right subtree at x. Obviously, ALG_L is a valid

strategy for the instance I_L. If S_L is the random variable denoting the total contribution of the omitted probes in an execution, then it is easy to see that $\mathbb{E}[OPT(G,t)] = \mathbb{E}[ALG_L] + \mathbb{E}[S_L]$. Similarly we define ALG_R. Here the algorithm ALG_R mimics the execution of $OPT(G,t)$ by replacing each $\alpha\beta$-probe, each t_α^{th} α-probe and each t_β^{th} β-probe by flipping a coin of suitable bias. As before it is easy to see that $\mathbb{E}[OPT(G,t)] = \mathbb{E}[ALG_R] + \mathbb{E}[S_R]$, where S_R is a random variable which denotes the total contribution of the probes omitted by ALG_R.

Before proceeding further, we introduce some definitions and notations. We use W_α to denote the contribution that a α-probe (if any) makes to the weight of the solution that $OPT(G,t)$ produces. We use $W_\alpha^{t_\alpha}$ to denote the contribution that a t_α^{th} α-probe (if it happens) makes. W_β and $W_\beta^{t_\beta}$ are similarly defined. We use $O_{\alpha\beta}$ to denote the event that $OPT(G,t)$ probes $\alpha\beta$; and $\overline{O_{\alpha\beta}}$ to denote the complement of event $O_{\alpha\beta}$. We also use $OPT_{\alpha\gamma}^{t_\alpha}$ ($\gamma \neq \alpha$) to denote the event that $OPT(G,t)$ probes $\alpha\gamma$ in the t_α-th α-probe. It follows that

$$\begin{aligned}\mathbb{E}[OPT] &= \mathbb{E}[ALG_R] + \mathbb{E}[S_R] \\ &= \mathbb{E}[ALG_R] + \mathbf{Pr}(O_{\alpha\beta})\mathbb{E}[S_R|O_{\alpha\beta}] \\ &+ \mathbf{Pr}(\overline{O_{\alpha\beta}})\left(\mathbb{E}[W_\alpha^{t_\alpha}|\overline{O_{\alpha\beta}}] + \mathbb{E}[W_\beta^{t_\beta}|\overline{O_{\alpha\beta}}]\right)\end{aligned} \tag{2.1}$$

$$\begin{aligned}\mathbb{E}[OPT] &= \mathbb{E}[ALG_L] + \mathbb{E}[S_L] \\ &= \mathbb{E}[ALG_L] + \mathbf{Pr}(O_{\alpha\beta})\mathbb{E}[S_L|O_{\alpha\beta}] \\ &+ \mathbf{Pr}(\overline{O_{\alpha\beta}})\left(\mathbb{E}[W_\alpha|\overline{O_{\alpha\beta}}] + \mathbb{E}[W_\beta|\overline{O_{\alpha\beta}}]\right)\end{aligned} \tag{2.2}$$

Multiplying (2.1) by $(1 - p_{\alpha\beta})$ and (2.2) by $p_{\alpha\beta}$ we get

$$\begin{aligned}
& \mathbb{E}[OPT] \\
= \; & p_{\alpha\beta}\mathbb{E}[ALG_L] + (1 - p_{\alpha\beta})\mathbb{E}[ALG_R] + \mathbf{Pr}(O_{\alpha\beta})\left(p_{\alpha\beta}\mathbb{E}[S_L|O_{\alpha\beta}] + (1 - p_{\alpha\beta})p_{\alpha\beta}w_{\alpha\beta}\right) \\
+ \; & \mathbf{Pr}(\overline{O_{\alpha\beta}})\left(p_{\alpha\beta}\mathbb{E}[W_\alpha|\overline{O_{\alpha\beta}}] + (1 - p_{\alpha\beta})\mathbb{E}[W_\alpha^{t_\alpha}|\overline{O_{\alpha\beta}}]\right) \\
+ \; & \mathbf{Pr}(\overline{O_{\alpha\beta}})\left(p_{\alpha\beta}\mathbb{E}[W_\beta|\overline{O_{\alpha\beta}}] + (1 - p_{\alpha\beta})\mathbb{E}[W_\beta^{t_\beta}|\overline{O_{\alpha\beta}}]\right) \qquad (2.3)
\end{aligned}$$

Auxilliary Graph J: Recall our assumption that $w_e \leqslant 1$ for each e. Now, for the sake of the analysis, we define an **auxiliary instance** J which is the same as the original input I except that edge weights are $z_e = 1 - x_e$ where $x_e = p_e w_e$ for each e. Define $p_{min} = \min_{e \in E} p_e$. The following observation plays a role in the lemmas that follow.

Observation 1. *For any edge $e \in E$, $w_e + \frac{z_e}{p_{min}} \leqslant \frac{1}{p_{min}}$.*

First, we obtain the following lemmas whose proofs are provided later.

Lemma 4.

$$\left(\frac{1 - x_{\alpha\beta}}{x_{\alpha\beta}}\right)\mathbb{E}(W_\alpha^{t_\alpha}(I)|\overline{O_{\alpha\beta}}) \;\leqslant\; \frac{\mathbb{E}(W_\alpha(J)|\overline{O_{\alpha\beta}})}{p_{min}}$$

Lemma 5.

$$p_{\alpha\beta}\mathbb{E}(W_\alpha(I)|\overline{O_{\alpha\beta}}) + (1 - p_{\alpha\beta})\mathbb{E}(W_\alpha^{t_\alpha}(I)|\overline{O_{\alpha\beta}}) \;\leqslant\; \frac{p_{\alpha\beta}}{p_{min}} \qquad (2.4)$$

An analogous inequality involving vertex β also holds.

$$p_{\alpha\beta}\mathbb{E}(W_\beta(I)|\overline{O_{\alpha\beta}}) + (1 - p_{\alpha\beta})\mathbb{E}(W_\beta^{t_\beta}(I)|\overline{O_{\alpha\beta}}) \;\leqslant\; \frac{p_{\alpha\beta}}{p_{min}} \qquad (2.5)$$

We observe that $w_{\alpha\beta}p_{\alpha\beta} \geqslant p_{min}$. Also we have $\mathbb{E}(S_L|O_{\alpha\beta}) \leqslant 2 \;\leqslant\; \frac{2w_{\alpha\beta}p_{\alpha\beta}}{p_{min}}$.

Theorem 2. *The greedy algorithm is a $\frac{2}{p_{min}^2}$-approximation algorithm.*

Proof : We prove the theorem by induction on the number of edges in G. The base cases of induction would be all those graphs G with $\mu(G) \leqslant 1$ where $\mu(G)$ is the maximum size (= the number of edges) of any matching in G. It is easy to verify that for each of the base cases, $Gr(G,t)$ is itself an optimal strategy. Our inductive hypothesis is that the greedy algorithm is a $\frac{2}{p_{min}^2}$-approximation to the optimal strategy for all graphs on a lesser number of edges. Using (5.3), (2.4) and (2.5), we have

$$\begin{aligned}
\mathbb{E}[OPT(I)] &\leqslant p_{\alpha\beta}\mathbb{E}[ALG_L] + (1-p_{\alpha\beta})\mathbb{E}[ALG_R] \\
&+ \mathbf{Pr}(O_{\alpha\beta})\left(p_{\alpha\beta}\mathbb{E}[S_L|O_{\alpha\beta}] + (1-p_{\alpha\beta})p_{\alpha\beta}w_{\alpha\beta}\right) + \mathbf{Pr}(\overline{O_{\alpha\beta}})\frac{2p_{\alpha\beta}^2 w_{\alpha\beta}}{p_{min}^2} \\
&\leqslant p_{\alpha\beta}\mathbb{E}[ALG_L] + (1-p_{\alpha\beta})\mathbb{E}[ALG_R] \\
&+ \mathbf{Pr}(O_{\alpha\beta})\left[\frac{2p_{\alpha\beta}^2 w_{\alpha\beta}}{p_{min}^2} + \frac{(1-p_{\alpha\beta})p_{\alpha\beta}w_{\alpha\beta}}{p_{min}^2}\right] + \mathbf{Pr}(\overline{O_{\alpha\beta}})\frac{2p_{\alpha\beta}^2 w_{\alpha\beta}}{p_{min}^2} \\
&\leqslant p_{\alpha\beta}\mathbb{E}[ALG_L] + (1-p_{\alpha\beta})\mathbb{E}[ALG_R] + \frac{2p_{\alpha\beta}^2 w_{\alpha\beta}}{p_{min}^2} + \frac{(1-p_{\alpha\beta})p_{\alpha\beta}w_{\alpha\beta}}{p_{min}^2} \\
&\leqslant p_{\alpha\beta}\mathbb{E}[ALG_L] + (1-p_{\alpha\beta})\mathbb{E}[ALG_R] + \frac{2p_{\alpha\beta}w_{\alpha\beta}}{p_{min}^2}
\end{aligned}$$

Using the last inequality and applying the inductive hypothesis to the smaller graphs, it follows that (with $OPT(I_L)$ ($OPT(I_R)$) representing the weight of the matching produced by an optimal strategy for I_L (I_R))

$$\begin{aligned}
\mathbb{E}[OPT(I)] &\leqslant p_{\alpha\beta}\mathbb{E}[OPT(I_L)] + (1-p_{\alpha\beta})\mathbb{E}[OPT(I_R)] + \frac{2p_{\alpha\beta}w_{\alpha\beta}}{p_{min}^2} \\
&\leqslant \frac{2p_{\alpha\beta}}{p_{min}^2}\mathbb{E}[Gr(I_L)] + \frac{2(1-p_{\alpha\beta})}{p_{min}^2}\mathbb{E}[Gr(I_R)] + \frac{2p_{\alpha\beta}w_{\alpha\beta}}{p_{min}^2} \\
&\leqslant \frac{2}{p_{min}^2}\left[p_{\alpha\beta}\mathbb{E}[Gr(I_L)] + (1-p_{\alpha\beta})\mathbb{E}[Gr(I_R)] + p_{\alpha\beta}w_{\alpha\beta}\right] \quad = \quad \frac{2}{p_{min}^2}\mathbb{E}[Gr(I)]
\end{aligned}$$

It now follows from the recursive definition of the performance of a strategy that the greedy strategy is a $\frac{2}{p_{min}^2}$ approximation to the optimal strategy. ∎

It only remains to prove Lemmas 4 and 5 and the proofs are presented below.

2.3.1 Proof of Lemma 4

$$\begin{aligned}
\frac{1-x_{\alpha\beta}}{x_{\alpha\beta}}\mathbb{E}[W_\alpha^{t_\alpha}(I)|\overline{O_{\alpha\beta}}] &= \sum_{\gamma\neq\beta}\frac{1-x_{\alpha\beta}}{x_{\alpha\beta}}w_{\alpha\gamma}p_{\alpha\gamma}\mathbf{Pr}(OPT_{\alpha\gamma}^{t_\alpha}|\overline{O_{\alpha\beta}}) \\
&\leqslant \sum_{\gamma\neq\beta}\frac{1-x_{\alpha\gamma}}{x_{\alpha\gamma}}w_{\alpha\gamma}p_{\alpha\gamma}\mathbf{Pr}(OPT_{\alpha\gamma}^{t_\alpha}|\overline{O_{\alpha\beta}}) \\
&\leqslant \frac{1}{p_{min}}\sum_{\gamma\neq\beta}(1-x_{\alpha\gamma})p_{\alpha\gamma}\mathbf{Pr}(OPT_{\alpha\gamma}^{t_\alpha}|\overline{O_{\alpha\beta}}) \\
&= \frac{\mathbb{E}[W_\alpha^{t_\alpha}(J)|\overline{O_{\alpha\beta}}]}{p_{min}} \leqslant \frac{\mathbb{E}[W_\alpha(J)|\overline{O_{\alpha\beta}}]}{p_{min}}
\end{aligned}$$

The first inequality follows since $\frac{1-x}{x}$ is a decreasing function of x in $(0, 1]$ and $x_{\alpha\beta}$ is the highest. ∎

2.3.2 Proof of Lemma 5

For each $\gamma \neq \beta$, let $E_{\alpha\gamma}$ denote the event that $\alpha\gamma$ is probed and the outcome is successful.

$$\begin{aligned}
\text{We have} \quad & p_{\alpha\beta}\mathbb{E}[W_\alpha(I)|\overline{O_{\alpha\beta}}] + (1-p_{\alpha\beta})\mathbb{E}[W_\alpha^{t_\alpha}(I)|\overline{O_{\alpha\beta}}] \\
\leqslant\ & p_{\alpha\beta}\mathbb{E}[W_\alpha(I)|\overline{O_{\alpha\beta}}] + (1-x_{\alpha\beta})\mathbb{E}[W_\alpha^{t_\alpha}(I)|\overline{O_{\alpha\beta}}] \\
\leqslant\ & p_{\alpha\beta}\mathbb{E}[W_\alpha(I)|\overline{O_{\alpha\beta}}] + \frac{x_{\alpha\beta}}{p_{min}}\mathbb{E}[W_\alpha(J)|\overline{O_{\alpha\beta}}] \text{ from Lemma 4} \\
\leqslant\ & p_{\alpha\beta}\mathbb{E}[W_\alpha(I)|\overline{O_{\alpha\beta}}] + \frac{p_{\alpha\beta}}{p_{min}}\mathbb{E}[W_\alpha(J)|\overline{O_{\alpha\beta}}] \\
=\ & p_{\alpha\beta}\left(\sum_{\gamma\neq\beta}\left(w_{\alpha\gamma}+\frac{1-x_{\alpha\gamma}}{p_{\min}}\right)\mathbf{Pr}\left(E_{\alpha\gamma}|\overline{O_{\alpha\beta}}\right)\right) \\
\leqslant\ & p_{\alpha\beta}\left(\sum_{\gamma\neq\beta}\frac{\mathbf{Pr}\left(E_{\alpha\gamma}|\overline{O_{\alpha\beta}}\right)}{p_{min}}\right) \leqslant \frac{p_{\alpha\beta}}{p_{min}}
\end{aligned}$$

The second last inequality follows from Observation 1. ∎

2.3.3 Proof of Observation 1

We have

$$w_e + \frac{1 - x_e}{p_{min}} \leqslant \frac{1 - x_e + w_e p_{min}}{p_{min}} \leqslant \frac{1 + w_e p_{min} - w_e p_e}{p_{min}} \leqslant \frac{1}{p_{min}}$$

The last inequality follows as $p_{min} \leqslant p_e$. ∎

2.3.4 Proof of Lemma 1

For each n, let G_n denote the graph $G = (V, E)$ where $V = \{u, v, a_1, \ldots, a_n, b_1, \ldots, b_n\}$ and $E = \{(u, v)\} \cup \{(u, a_i) : 1 \leqslant i \leqslant n\} \cup \{(v, b_i) : 1 \leqslant i \leqslant n\}$. Let $w_{uv} = W$ and $p_{uv} = 1 - \frac{1}{n}$. Let $p = p_{min} = \frac{1}{\sqrt{n}}$ and define W' by $W'p = W(1 - 1/n)^2$. Let $p_e = p$ and $w_e = W'$ for every $e \neq uv$. Let u and v be both have a patience parameter of $n + 1$ and let each of a_i's and b_i's have a patience parameter of 1. The expected weight of the solution produced by the greedy algorithm can be shown to be at most $W(1 - \frac{1}{n}) + 2W'(1 - (1 - p)^n)/n \leqslant W\left(1 - \frac{1}{n} + \frac{2}{\sqrt{n}}\right) = W[1 + o(1)]$. Now consider the strategy which first probes each of the n edges (u, a_i) and then probes each of the n edges (v, b_i) and then probes uv. The expected weight of the solution of this strategy is at least $2W'(1 - (1 - p)^n) = 2W\sqrt{n}[1 - o(1)]$. ∎

2.4 A vertex-wise greedy variant

GRD-VW is one variant that naturally comes to one's mind and this also does not possess a good approximation ratio. This variant tries to be greedy vertex-wise. That is, it first computes for each vertex v a value m_v which is computed as follows. Let $\sigma = (e_1, \ldots, e_{t_v})$ be an optimal ordering (sorted in decreasing weights w_e) of the t_v heaviest (in terms of expected individual contributions $w_e p_e$ one obtains if probed) edges incident and available (for probing) at v. m_v denotes the expected contribution one obtains by probing edges as

per σ. It can be easily computed using the expression provided below. **GRD – VW** then chooses a vertex u for which $m_u = \max_v m_v$ for probing incident edges. Here, t_v and d_v are the current values of v's patience and its degree. It can be verified that $m_v = \sum_{i \leqslant t_v} w_i p_i \left(\prod_{j<i} 1 - p_j \right)$. A formal description of the algorithm is presented below. As before, the graph contains only edges joining vertices with positive patience values.

Algorithm 2 GRD-VW $MG_r(G, t)$:

$E' \leftarrow E$. $M \leftarrow \emptyset$.
while $E' \neq \emptyset$ **do**
    Choose any vertex u which maximizes m_v
    Let $\sigma_u = (e_1, \ldots, e_{t_u})$, $e_j = (uv_j)$, denote an optimal order of edges available for probing.
    $j \leftarrow 1$.
    while $j \leqslant t_u$ and $t_u > 0$ **do**
        Probe e_j and add e_j to M if e_j is found to be present.
        If $e_j \in M$, then set each of t_u and t_{v_j} to be zero; **else** decrement t_u and t_{v_j}.
        Remove e_j from E'. Increment j.
        Remove any edge in E' incident at u (v_j) if t_u (t_{v_j}) equals zero.
        Rationalize(G, t).
    endwhile
endwhile
Output M.

The following theorem establishes a lower bound on the worst-case approximation ratio of the greedy variant $MGr(G, t)$ thereby establishing that the approximation ratio can become unbounded even if we restrict ourselves only to unweighted instances. This is in contrast to the edge-wise greedy heuristic which was shown to have an approximation ratio of 2 for unweighted instances.

Lemma 6. *There exists an infinite and explicit family* $\{(G_n, t_n)\}_{n \geqslant 1}$ *of unweighted input instances such that the expected size of the solution obtained by* $MGr(G_n, t_n)$ *is smaller than that of an optimal strategy by a multiplicative factor of nearly* $\Omega\left(\frac{1}{p_{max}}\right)$ *where* $p_{max} = \max_e p_e$.

Proof of Lemma 6 : For each n, let G_n denote the graph $G = (V, E)$ where

$$V = \{u, a_1, \ldots, a_n, b_1, \ldots, b_n\}; \quad E = \{(u, a_i) : 1 \leqslant i \leqslant n\} \cup \{(a_i, b_i) : 1 \leqslant i \leqslant n\}.$$

Let $p = p(n)$ be any function such that $p \to 0$ and $p = \omega(\frac{1}{n})$. Define $q = q(n) := \frac{2p}{n}$. Also, let $p_{(u,a_i)} = q$ for each i, and $p_{(a_i,b_i)} = p$ for each i. We note that $p_{max} = p$. Let u have a patience parameter of n and let each of a_i's and b_i's have a patience parameter of 1. Consider the strategy which probes each of the n edges (a_i, b_i) and outputs the resulting matching. The expected size of the solution to this strategy is exactly np. Hence the expected size of any optimal strategy is at least np.

We now analyze $MGr(,)$. Notice that

$$m_u = 1 - (1-q)^n = nq - \Theta((nq)^2) = 2p - \Theta(p^2)$$

and $m_{a_i} = m_{b_i} = p$ for each i. Hence $m_u > m_v$ for each $v \neq u$. Without loss of generality, assume that $MGr(,)$ probes edges in the order $(ua_1, \ldots, ua_n)$. Using MGr to denote the size of the solution produced by $MGr(G, t)$, we have

$$\begin{aligned}
\mathbb{E}[MGr] &= \sum_{j=0}^{n-1} (1-q)^j q\,(1 + (n-j-1)p) \\
&= 1 - (1-q)^n + \sum_{j=0}^{n-1} (n-j-1)(1-q)^j pq \\
&= 1 - (1-q)^n + pq(1-q)^{n-1} \left(\sum_{j=0}^{n-1} j(1-q)^{-j} \right) \\
&= 1 - (1-q)^n + pq(1-q)^{n-1} \left(\frac{(1-q)^{-1} - n(1-q)^{-n} + (n-1)(1-q)^{-n-1}}{q^2(1-q)^{-2}} \right) \\
&= 1 - (1-q)^n + p \left(\frac{(1-q)^n - n(1-q) + (n-1)}{q} \right) \\
&= 1 - (1-q)^n + p \left(\frac{1 - nq + \Theta((nq)^2) - n + nq + n - 1}{q} \right) \\
&= 2p - \Theta(p^2) + \frac{n}{2} \cdot \Theta(p^2) \;=\; \Theta(np^2)
\end{aligned}$$

Hence the ratio $\frac{\mathbb{E}[OPT(G,t)]}{\mathbb{E}[MGr]} = \Omega(p^{-1})$ where $p = p_{max}$. This establishes the lemma. ∎

2.5 A generalized greedy variant

Proof of Lemma 3 : For each n, let G_n denote the graph defined in the proof of Lemma 6 with the same patience values and edge probabilities except that we redefine p and q as follows. Define $p = p(n) := \frac{k^\epsilon}{n}$. It follows that $p \to 0$ and $p = \omega(\frac{1}{n})$. Define $q = q(n) := \frac{2p}{k}$. It follows that $nq \to 0$. As shown before, the expected size of any optimal strategy is at least np.

We now analyze $Gr_k(,)$. Recall our assumption that k divides n. Notice that

$$m_u = 1 - (1-q)^k = kq - \Theta((kq)^2) = 2p - \Theta(p^2)$$

and $m_{a_i} = m_{b_i} = p$ for each i. Hence $m_u > m_v$ for each $v \neq u$, as long as u has at least k un-probed edges incident at it and hence $Gr_k()$ will pick k of these edges and probe them consecutively. Since k divides n, this means that $Gr_k(,)$ will probe all edges incident at u and stop with that. Without loss of generality, assume that $Gr_k(,)$ probes edges in the order $(ua_1, \ldots, ua_n)$. Using Gr_k to denote the size of the solution produced by $Gr_k(G,t)$, we have (as shown before)

$$\begin{aligned}
\mathbb{E}[Gr_k] &= 1 - (1-q)^n + p\left(\frac{(1-q)^n - n(1-q) + (n-1)}{q}\right) \\
&= 1 - (1-q)^n + p\left(\frac{1 - nq + \Theta((nq)^2) - n + nq + n - 1}{q}\right) \\
&= nq - \Theta((nq)^2) + \frac{k}{2} \cdot \Theta(k^{-2+2\epsilon}) \;=\; \Theta(k^{-1+2\epsilon})
\end{aligned}$$

Hence the ratio $\frac{\mathbb{E}[OPT(G,t)]}{\mathbb{E}[Gr_k]} = \Theta(\frac{np}{k^{-1+2\epsilon}}) = \Theta(k^{1-\epsilon}) \to \infty$ as $n \to \infty$. This establishes the lemma. ∎

2.6 Remarks

We analyzed some variants of greedy heuristic for both weighted and unweighted stochastic matching instances. The following observations are relevant in this context and the last question should be addressed to gather a better comprehension of greedy heuristics.

- For the greedy heuristic $Gr(,)$ applied to weighted instances, the upper and lower bounds on the worst-case approximation ratio still differ by a multiplicative factor of $\frac{1}{p_{min}}$. It would be interesting to reduce this gap and obtain a tight upper bound on the worst-case ratio.

- The assumption that k divides n can be weakened to $n \pmod k = 0$ or $n \pmod k \geqslant (\frac{1}{2} + \delta)k$ for some fixed $\delta > 0$.

- The multiplicative factor $\Theta(k^{1-\epsilon})$ in the statement of Lemma 3 can be improved to $\Theta(\frac{k}{\omega})$ where $\omega = \omega(n)$ is any sufficiently slow-growing function satisfying $\omega = o(k)$.

- The assumption of $k \to \infty$ in the statement of Lemma 3 can be removed with a corresponding replacement of the term $\Theta(k^{1-\epsilon})$ by a suitable function $f(k)$ (where $f(k) \to \infty$ obviously if $k \to \infty$). This establishes that Gr_k is worse than an optimal strategy by a factor of at least $f(k)$.

- Does there exist (for every fixed $k(n)$), a function $g(k)$ such that $Gr_k(,)$ produces a solution whose expected size is within a multiplicative factor of $g(k)$ from that of an optimal solution (for all instances). In particular, we conjecture that for every $c \geqslant 1$, there exists a value $g(c)$ such that $Gr_c(,)$ is a $g(c)$-approximation algorithm for unweighted instances. We know that $g(1) = 2$ from the work of [Ada11].

Chapter 3

Approximation algorithm for Online Stochastic Matching

3.1 Introduction

Bansal et al. [BGL+10] introduced this online version. It models the sale of items from a set A to buyers arriving in an online fashion. Each buyer has to be processed before we consider the next arriving buyer. The processing of each buyer involves showing a select subset of items in some order until the buyer likes an item (if it happens) in which case both the item and the buyer are removed from the picture. To each buyer, we can associate a type/profile and the type characterizes (i) the patience t_b, (ii) probability p_{ab} that a buyer of type b buys item a, and (iii) w_{ab} the revenue generated if it happens. The type of each arriving buyer is independently and identically distributed over the set B of types. Here, the buyers arrive online. The number of buyers that are going to arrive is known to the algorithm. The goal is to design an efficient online algorithm which produces a matching whose expected revenue is as large as possible. The performance of the algorithm is compared with the expected revenue from the matching produced by an optimal strategy. [BGL+10] presents a 7.92-approximate algorithm for this problem. We

propose and analyze a 5.2-approximate algorithm for the same problem.

3.2 Preliminaries

We use a randomized rounding procedure in designing the approximation algorithm for online stochastic matching. This randomized rounding procedure was introduced by Gandhi et al. [GKPS06]. We call it GKPS rounding for short. We describe below the main properties of GKPS rounding scheme. We denote by $\partial(u)$ the set of edges incident on vertex u.

GKPS Rounding Scheme: We list below the main properties of the dependent rounding algorithm given by Gandhi et al [GKPS06] that will be useful for the present problem.

Theorem 3. *[[GKPS06]] Let $(A, B; E)$ be a bipartite graph and $z_e \in [0, 1]$ be fractional values for each edge $e \in E$. The GKPS algorithm is a polynomial-time randomized procedure that outputs values $Z_e \in \{0, 1\}$ for each edge $e \in E$ such that the following properties hold:*

P1. Marginal Distribution: For every edge e, $Pr[Z_e = 1] = z_e$.

P2. Degree Preservation: For every vertex $u \in A \cup B$, $\sum_{e \in \partial(u)} Z_e \in \{\lfloor \sum_{e \in \partial(u)} z_e \rfloor, \lceil \sum_{e \in \partial(u)} z_e \rceil\}$.

P3. Negative correlation: For any vertex u and any set of edges $S \subseteq \partial(u)$: $Pr[\bigwedge_{e \in S} (Z_e = 1)] \leqslant \prod_{e \in S} Pr[Z_e = 1]$.

3.3 Approximation algorithm

The Online Stochastic Matching problem models the problem of maximizing the revenue from the sales of a set of items to buyers coming in an online fashion. This problem was introduced and studied by Bansal et al. in [BGL+10]. The problem is described below. Let $(G = (A, B, A \times B), \{p_{ab}\}_{a \in A, b \in B}, \{e_b\}_{b \in B})$ be the random model underlying actual online arrival of the buyers and their preferences. The additional input to the algorithm

is $(\{w_{ab}\}_{a\in A,b\in B}, \{t_b\}_{b\in B})$. A is a set of items with exactly one copy of each item and B is a set of buyer types/profiles. For each buyer of type $b \in B$ and item a, p_{ab} denotes the probability that such a buyer buys the item a and w_{ab} denotes the revenue generated if a is sold to this buyer. Each buyer of type b has a patience for at most t_b probes, that is, she will consider at most t_b distinct items shown one by one. The buyer buys the first item she likes or leaves without buying any item. There are n actual buyers who arrive and the type of each buyer is identically and independently distributed over B with e_b denoting the expected number of buyers of type b and $\sum_b e_b = n$. As in [BGL+10], by duplicating buyer types, we assume without loss of generality that there are n different buyer types and the expected number of buyers of each type is 1. Let the actual graph that defines the input for a particular run of the algorithm be $\hat{G} = (A, \hat{B}, A \times \hat{B})$. In this graph the probability associated with any edge $(a, \hat{b})$ is p_{ab} and weight is w_{ab} provided $\hat{b}$ belongs to type b. An optimal algorithm for this problem is an adaptive strategy (possibly randomized) for probing the edges incident at arriving buyers, for which the expected revenue is maximum. We denote this maximum by $OPT(G)$, or shortly OPT.

To maximize the expected revenue from $\hat{G}$ we solve the LP written below for the expected graph G as defined before. Then we use the LP solution to guide the choice (of items to be shown) for the first buyer of each type and ignore the later arrivals of buyers of same type.

$$\begin{aligned}
\text{maximize} \quad & \sum_{a\in A, b\in B} x_{ab} \cdot w_{ab} \quad \text{subject to} \\
& \sum_{a\in A} x_{ab} \leqslant 1 \ \forall b \in B \\
& \sum_{b\in B} x_{ab} \leqslant 1 \ \forall a \in A \\
& \sum_{a\in A} y_{ab} \leqslant t_b \ \forall b \in B \\
& x_{ab} = p_{ab} \cdot y_{ab}; \quad y_{ab} \in [0,1] \ \forall a \in A, \quad \forall b \in B
\end{aligned} \tag{3.1}$$

Let $LP(G)$ denote the optimal value of the above LP. The following bound was established

in [BGL+10].

Lemma 7. *[Lemma 11, Lemma 13 of [BGL+10]] $OPT(G) \leqslant LP(G)$.*

We combine some of the salient features (like GKPS rounding) of the offline algorithm with some salient features (like ignoring 2nd or later arrivals of any buyer type) of the online algorithm of Bansal et al., [BGL+10] to get a new algorithm (see Algorithm 2) for the online stochastic matching problem. This also required us to introduce a new ordering which combines the random ordering of online arrivals with a chosen random ordering of items. Analyzing the new algorithm, we obtain the following improved result. This improves the approximation ratio from the previous one (from [BGL+10]) of 7.92.

Theorem 4. *There exists an adaptive and randomized strategy for the online stochastic matching problem which produces a matching whose expected cost is at least $LP(G)/5.2$. Hence, we get a 5.2- approximation algorithm for this problem.*

Algorithm 3

Choose uniformly a random ordering τ of the items in A.
$(x, y) \leftarrow$ optimal solution of the LP on the expected graph G.
$\hat{y} \leftarrow$ round y to an integral solution using GKPS rounding.
$\hat{E} \leftarrow \{e|\hat{y}_e = 1\}$.
When any buyer $\hat{b}$ (of type b) arrives **do**
if $\hat{b}$ is the first arrival of type b **then**
    One by one offer (as per τ) each item $i \in \{a|(a, b) \in \hat{E}\}$ that is still unsold until either an item is bought by $\hat{b}$ or its patience is exhausted.
else
    Ignore $\hat{b}$.
end if

Notations : Throughout this section, we employ the following notations (with the stated meanings) for the sake of keeping the mathematical expressions simpler. Let A denote an event, ω a random choice and Y a random variable. By $\mathbb{E}_A(Y)$ we mean the conditional expectation $\mathbb{E}[Y|A]$. By $\mathbb{E}_\omega(Y)$, we mean the expectation of $Y = Y(\omega)$ with respect to the choice ω.

First, we give an informal description of our algorithm. It combines ideas from the approximation algorithms (proposed by Bansal et al. in [BGL+10]) for both the offline and

online stochastic matching problems. We initially choose uniformly randomly an ordering τ of all items. After solving the above stated LP, we apply the randomized GKPS rounding procedure [GKPS06] (which is described below) to obtain an integral solution (the set of edges which are likely to be probed). Let $\hat{E}$ denote this set of edges from E. As in [BGL+10], we focus only on the first buyer of any type and ignore later buyers of the same type. For each first arrival of a buyer of type b, we try to probe edges from $\hat{E}$ that are incident at b as per the order τ of the corresponding items until either an item is bought by the buyer or its patience is exhausted.

Let $B' \subseteq B$ denote the random subset of buyer types represented at least once in the actual online arrivals of buyers. Conditioned on a given value of B', the order η induced by the first buyers of different types $b \in B'$ is uniform over B'. We combine η over (buyer types) and τ (over items) to define a lexicographic order ν (first compare with buyer types and then with items) over edges of $\hat{E}$. ν will play a role in bounding (from below) the expected revenue that the first arrival of a buyer of type b contributes. Note that τ and η are independent of each other.

Given $e = (i, b)$, B' and $\hat{E}$ such that $b \in B'$ and $e \in \hat{E}$, we use $\partial_{\hat{E}}(b, e)$ to denote the set of edges $f \in \hat{E}$ which are also incident at b. Similarly, we use $\partial_{\hat{E}}(i, e)$ to denote the set of edges $f \in \hat{E}$ involving types from B' and which are also incident at i. We use $\partial_{\hat{E}}(e)$ to denote the union $\partial_{\hat{E}}(b, e) \cup \partial_{\hat{E}}(i, e)$. Let $B(e, \nu) \subseteq \partial_{\hat{E}}(e)$ denote the set of those edges which precede e in the ordering ν. Also, let $B(e, \eta)$ denote the set of those edges in $B(e, \nu)$ which are incident at i. Similarly, we let $B(e, \tau)$ denote the set of edges from $B(e, \nu)$ which are incident at b. For any particular type b, we denote by A_b the event that a buyer of type b arrives at least once. We first obtain a lower bound on $\mathbf{Pr}[e \text{ is probed } \mid e \in \hat{E}, A_b]$ as stated in the following lemma.

Lemma 8. *For an arbitrary type b and an arbitrary edge e incident at b,*

$$\mathbf{Pr}\left(e \text{ is probed} \mid e \in \hat{E}, A_b\right) \geqslant \mathbb{E}_{b \in B', e \in \hat{E}}\left[\mathbb{E}_{\nu}\left[\prod_{f \in B(e,\nu)} (1 - p_f) \mid B', \hat{E}\right]\right].$$

Proof. Given a choice of B' and $\hat{E}$ such that $b \in B'$ and $e \in \hat{E}$, e will be probed if, for each $f \in B(e,\nu)$, f is absent (irrespective of whether f was probed or not). Therefore $Pr[e \text{ is probed} \mid B', \hat{E}] \geqslant \mathbb{E}_\nu[\prod_{f \in B(e,\nu)} (1 - p_f) \mid B', \hat{E}]$. Now, considering expectation over the random choices determining B' and $\hat{E}$, we obtain the desired inequality. □

Before analyzing the new ordering ν, we introduce some definitions and some useful facts established in [BGL+10].

Definition 2. *Let r and p_{max} be positive real values. Denote by $\eta(r, p_{max})$ the minimum value of $\prod_{i=1}^{t}(1 - p_i)$ subject to the constraints $\sum_{i=1}^{t} p_i \leqslant r$ and $0 \leqslant p_i \leqslant p_{max}$ for $i = 1 \ldots, t$. Also, let $\rho(r, p_{max})$ be defined by $\rho(r, p_{max}) = \int_0^1 \eta(xr, xp_{max})\,\mathrm{d}x$.*

Lemma 9. *[Lemma 5, Lemma 7 of [BGL+10]] Let r and p_{max} be positive real values. Then,*

1. $\eta(r, p_{max}) = (1 - p_{max})^{\lfloor \frac{r}{p_{max}} \rfloor}(1 - (r - \lfloor \frac{r}{p_{max}} \rfloor p_{max})) \geqslant (1 - p_{max})^{(\frac{r}{p_{max}})}$.
2. $\rho(r, p_{max})$ *is convex and decreasing on* r.
3. $\rho(r, p_{max}) \geqslant \frac{1}{r+p_{max}}(1 - (1 - p_{max})^{1+\frac{r}{p_{max}}}) \geqslant \frac{1}{r+p_{max}}(1 - e^{-r})$.

The following lemma follows from the proof arguments of Lemma 6 of [BGL+10].

Lemma 10. *For any edge $e = (i, b)$ and for every given B' and $\hat{E}$ such that $b \in B'$ and $e \in \hat{E}$, let σ be a random ordering of edges in $\partial_{\hat{E}}(b, e)$ (or $\partial_{\hat{E}}(i, e)$). Let $p_{max} = max_{f \in E} p_f$. Let $B(e, \sigma)$ denote the set of edges in $\partial_{\hat{E}}(b, e)$ (or $\partial_{\hat{E}}(i, e)$) which precede e in σ. Let r be defined by $r = \sum_{f \in \partial_{\hat{E}}(b,e)} p_f$ (or $r = \sum_{f \in \partial_{\hat{E}}(i,e)} p_f$). Then $\mathbb{E}_\sigma[\prod_{f \in B(e,\sigma)} (1 - p_f) \mid B', \hat{E}] \geqslant \int_0^1 \eta(xr, xp_{max})\,\mathrm{d}x$.*

Using the above lemma, the following lemma analyzes and establishes a corresponding lower bound on the expectation (with respect to the new ordering ν).

Lemma 11. *For any edge $e = (i, b)$ and for every given B' and $\hat{E}$ such that $b \in B'$ and $e \in \hat{E}$, $\mathbb{E}_\nu\left[\prod_{f \in B(e,\nu)}(1 - p_f) \mid B', \hat{E}\right] \geqslant \rho(r_1, p_{max})\rho(r_2, p_{max})$ where*
$r_1 = \sum_{f \in \partial_{\hat{E}}(i,e)} p_f$ and $r_2 = \sum_{f \in \partial_{\hat{E}}(b,e)} p_f$.

Proof. We have

$$\begin{aligned}
\mathbb{E}_\nu\left[\prod_{f\in B(e,\nu)}(1-p_f)\mid B',\hat{E}\right] &= \mathbb{E}_\nu\left[\left(\prod_{f\in B(e,\eta)}(1-p_f)\right)\left(\prod_{f\in B(e,\tau)}(1-p_f)\right)\mid B',\hat{E}\right] \\
&= \mathbb{E}_\eta\left[\prod_{f\in B(e,\eta)}(1-p_f)\mid B',\hat{E}\right]\cdot\mathbb{E}_\tau\left[\prod_{f\in B(e,\tau)}(1-p_f)\mid B',\hat{E}\right] \\
&\geqslant \rho(r_1,p_{max})\cdot\rho(r_2,p_{max})
\end{aligned}$$

In the above derivation, the second equality follows from the observation that the random orderings of edges of $\partial_{\hat{E}}(i,e)$ and $\partial_{\hat{E}}(b,e)$ are independent and depend respectively only on the orderings η and τ. The inequality follows from applying Lemma 10 to the two expectations in the previous line. □

We also need the following lemma.

Lemma 12. *$\rho(r_1,p_{max})\cdot\rho(r_2,p_{max})$ is convex.*

Proof. We know that the product of two nonincreasing convex functions on $\mathbb{R}$ is a convex function on $\mathbb{R}$ [BV04]. We know from Lemma 9 that both $\rho(r_1,p_{max})$ and $\rho(r_2,p_{max})$ are convex and decreasing. Hence $\rho(r_1,p_{max})\cdot\rho(r_2,p_{max})$ is convex. □

The following lemma is similar to Lemma 9 of [BGL+10] and plays a role in our analysis. The asymptotics $o(1)$ is with respect to n.

Lemma 13. *For some $\epsilon=\epsilon(n)=o(1)$, for every sufficiently large n, for every edge $e=(i,b)$ in the expected graph,*

$$\mathbb{E}_{b\in B',e\in\hat{E}}\left[\sum_{f\in\partial_{\hat{E}}(i,e)}p_f\right] \leqslant \left(1-\frac{1}{e}+\epsilon\right) \tag{3.2}$$

$$\mathbb{E}_{b\in B',e\in\hat{E}}\left[\sum_{f\in\partial_{\hat{E}}(b,e)}p_f\right] \leqslant 1 \tag{3.3}$$

Proof. We only prove Inequality (3.2).

$$\begin{aligned}
\mathbb{E}_{b\in B',e\in\hat{E}}\left[\sum_{f\in\partial_{\hat{E}}(i,e)} p_f\right] &= \sum_{f=(i,b')} \mathbf{Pr}(b'\in B' \mid b\in B')\cdot\mathbf{Pr}[\hat{y}_f=1\mid \hat{y}_e=1]\cdot p_f \\
&\leqslant \left(1-\frac{1}{e}+\epsilon\right)\cdot\left(\sum_{f=(i,b')}\mathbf{Pr}[\hat{y}_f=1]\cdot p_f\right) \text{ by P3 of Theorem 3} \\
&= \left(1-\frac{1}{e}+\epsilon\right)\cdot\left(\sum_{f=(i,b')} y_f\cdot p_f\right) \text{ by P1 of Theorem 3} \\
&\leqslant \left(1-\frac{1}{e}+\epsilon\right) \text{ by Inequality (3.1)}
\end{aligned}$$

Similarly Inequality (3.3) is proved. □

In the next lemma, we analyze the performance of Algorithm 2 with respect to the objective value of $LP(G)$.

Lemma 14. *For some $\epsilon=\epsilon(n)=o(1)$, the expected revenue of Algorithm 2 is at least $(1-\frac{1}{e})\cdot\rho(1-\frac{1}{e}+\epsilon,p_{max})\cdot\rho(1,p_{max})\cdot LP(G)$.*

Proof. We use some of the notations from [BGL+10]. For any type b, let R_b denote the revenue generated by the first buyer (if any) of type b. The algorithm ignores later buyers of this type and hence gets no revenue from these buyers. We have

$$\mathbb{E}[R_b] = \mathbb{E}[R_b\mid A_b]\cdot\mathbf{Pr}(A_b) \geqslant \left(1-\frac{1}{e}\right)\mathbb{E}[R_b\mid A_b] \tag{3.4}$$

$$\begin{aligned}
\mathbb{E}[R_b\mid A_b] &= \sum_{a\in A} w_{ab}\cdot p_{ab}\cdot\mathbf{Pr}(ab \text{ is probed } \mid A_b) \\
&= \sum_{a\in A} w_{ab}\cdot p_{ab}\cdot\mathbf{Pr}(ab\in\hat{E}\mid A_b)\cdot\mathbf{Pr}(ab \text{ is probed } \mid A_b, ab\in\hat{E}) \\
&= \sum_{a\in A} w_{ab}\cdot p_{ab}\cdot y_{ab}\cdot\mathbf{Pr}(ab \text{ is probed } \mid A_b, ab\in\hat{E}) \\
&= \sum_{a\in A} w_{ab}\cdot x_{ab}\cdot\mathbf{Pr}(ab \text{ is probed } \mid A_b, ab\in\hat{E})
\end{aligned} \tag{3.5}$$

where we have (by applying Lemma 8)

$$\begin{aligned}
\mathbf{Pr}\left(ab \text{ is probed } \mid A_b, ab \in \hat{E}\right) &\geqslant \mathbb{E}_{b\in B', ab\in\hat{E}}\left[\mathbb{E}_{\nu}\left[\prod_{f\in B(e,\nu)}(1-p_f) \mid B', \hat{E}\right]\right] \\
&\geqslant \mathbb{E}_{b\in B', ab\in\hat{E}}\left[\rho(r_1, p_{max})\cdot\rho(r_2, p_{max})\right] \text{ applying Lemma 11} \\
&\geqslant \rho\left(\mathbb{E}\left[r_1 \mid b \in B', ab \in \hat{E}\right], p_{max}\right)\cdot \\
&\quad \rho\left(\mathbb{E}\left[r_2 \mid b \in B', ab \in \hat{E}\right], p_{max}\right) \\
&\geqslant \rho\left(1-\frac{1}{e}+\epsilon, p_{max}\right)\cdot\rho\left(1, p_{max}\right) \qquad (3.6)
\end{aligned}$$

In this derivation, we have applied multivariate Jensen's inequality for convex functions and also Statement (2) of Lemma 9. Combining equalities and inequalities 3.4, 3.5 3.6, we obtain the following lower bound on the expected revenue produced by the algorithm.

$$\begin{aligned}
\sum_{b\in B}\mathbb{E}[R_b] &\geqslant \left(1-\frac{1}{e}\right)\cdot\rho\left(1-\frac{1}{e}+\epsilon, p_{max}\right)\cdot\rho\left(1, p_{max}\right)\cdot\left(\sum_{a\in A, b\in B} w_{ab}\cdot x_{ab}\right) \\
&= \left(1-\frac{1}{e}\right)\cdot\rho\left(1-\frac{1}{e}+\epsilon, p_{max}\right)\cdot\rho\left(1, p_{max}\right)\cdot LP(G)
\end{aligned}$$

□

From Lemma 7 and Lemma 14, we get a $(1-\frac{1}{e})\cdot\rho(1-\frac{1}{e}+\epsilon, p_{max})\cdot\rho(1, p_{max})$-factor approximation for the online stochastic matching problem. We note here that the worst case approximation ratio which occurs at $p_{max} = 1$ is at most 5.2 and this establishes Theorem 4.

Chapter 4

Approximation of MIS for B_1-VPG graphs

In this chapter, we present an efficient approximation MIS algorithm for B_1-VPG graphs, with an improved $O((\log n)^2)$ approximation guarantee. It applies the divide-and-conquer paradigm to reduce the given instance into three subinstances and recursively solves each of them.

4.1 Preliminaries

The reason why we focus only on intersection graphs formed by geometric objects of shape "L" is that the other three shapes can be obtained by rotating the plane by $90, 180$ and 270 degrees in the clockwise direction. The four shapes are denoted by $\llcorner, \lrcorner, \ulcorner, \urcorner$. Henceforth, we use l to denote a geometric object with one of these four shapes. Also, for ease of further discussion, we specifically use L to denote a geometric object with shape $\llcorner$.

In view of the rotational symmetries, any algorithm which solves MIS (exactly or ap-

proximately) over L-graphs can also be suitably adapted to solve MIS (with the same performance guarantee) over B_1-VPG graphs. We get the following as a corollary :

Lemma 15. *If there exists an efficient algorithm A for solving MIS over L-graphs with a performance guarantee bounded by $\alpha(n)$, then there exists an efficient algorithm B for solving MIS over B_1-VPG graphs, with a performance guarantee at most $4\alpha(n)$. Here, n stands for the size of the input for both algorithms.*

Proof. The algorithm B works as follows. Given a B_1-VPG graph $G = (V, E)$, we decompose G into four induced subgraphs $G_1, \ldots, G_4$ formed by objects of each specific shape. We apply algorithm A to G_j to get an approximate MIS I_j, for each j. Algorithm B then outputs any I_l such that $|I_l| = \max_j |I_j|$.

If I^* denotes a MIS in G and I_j^* denotes a MIS in G_j (for each j), then it follows that $\max_j |I_j^*| \geqslant \max_j |I^* \cap V(G_j)| \geqslant |I^*|/4$. If I_j denotes the approximate solution obtained A for G_j, then $|I_j| \geqslant |I_j^*|/\alpha(n)$ and hence $\max_j |I_j| \geqslant |I^*|/(4\alpha(n))$. □

This lemma explains why it suffices to focus only on L-graphs. Henceforth, for the rest of this chapter, we focus only on L-graphs.

The intersection point of the two sides of an L is defined as the *corner* of the L and is denoted by c_L, the tip of the horizontal arm is denoted by h_L and that of the vertical arm is denoted by v_L. For an object L, we use (cx, cy, hx, vy) to denote respectively the x- and y- coordinates of c_L, the x- coordinate of h_l and the y- coordinate of v_l. This 4-tuple completely describes L. The set of points constituting L is denoted by P_L and is given by $P_L = \{(x, cy) \ : \ cx \leqslant x \leqslant hx\} \cup \{(cx, y) \ : \ cy \leqslant y \leqslant vy\}$. We say that two distinct objects L_1 and L_2 intersect if $P_{L_1} \cap P_{L_2} \neq \emptyset$. L_1 and L_2 are said to be *independent* if and only if they do not intersect. A set of L's such that no two of them forms an intersecting pair is said to be an *independent* set. Suppose two objects L_1 and L_2 are such that $L_1.cx < L_2.cx$ and $L_1.cy < L_2.cy$. Then we say that $c_{L_1} < c_{L_2}$.

When the length of the vertical side of an L is equal to the horizontal side of an L we say that it is equilateral. Since for equilateral L's the length of the horizontal side is equal to that of the vertical side, we simply use $le(L)$ to denote the length of the horizontal side as well as the vertical side. All logarithms used below are with respect to base 2. We denote a set $\{1, 2, \ldots, n\}$ by $[n]$.

4.2 MIS Approximation over L-Graphs

Maximum Independent Set in L-graphs
Input: A set $\mathcal{S}$ of L's
Output: a set $I \subset \mathcal{S}$ such that I is independent and $\|I\|$ is maximized.

The decision version of this problem is NP-complete (see Theorem 7). The decision version corresponds to determining, given a L-graph G and an integer $k \geqslant 1$, if G has an independent set of size k. Below, we present approximation algorithms for the optimization version stated before.

Before proceeding further, for the sake of keeping the arguments simple, we introduce an assumption which is stated in the following claim and which is formally justified in the next chapter.

Claim 1. *Without loss of generality, we can assume that*

(*i*) $L_1.cx \neq L_2.cx$ *and* $L_1.cy \neq L_2.cy$ *for any pair of distinct* $L_1, L_2 \in \mathcal{S}$*;*

Definition 3. *For a sorted sequence* $x_1 < x_2 < \ldots < x_n$ *of distinct reals, we define its median to be the* $x_{\frac{n+1}{2}}$ *if* n *is odd or the average of* $x_{\frac{n}{2}}$ *and* $x_{\frac{n}{2}+1}$ *if* n *is even.*

Our approach is broadly to divide and conquer. We sort the objects in $\mathcal{S}$ in increasing order of their $L.cx$ values. Define x_{med} to be the median of this sorted order. Then, we compute the sets S_1, S_2 and S_{12} defined as follows.

$S_1 := \{L \in \mathcal{S} \ : \ L.hx < x_{med}\}$.

$S_2 := \{L \in \mathcal{S} \ : \ L.cx > x_{med}\}$.

$S_{12} := \{L \in \mathcal{S} \ : \ L.cx \leqslant x_{med} \leqslant L.hx\}$.

The sets S_1, S_2 and S_{12} form a partition of $\mathcal{S}$. Also, any pair of $L_1 \in S_1, L_2 \in S_2$ are independent. The problem is solved by applying the recursive Algorithm *IndSet*1 presented below. This algorithm (on input $\mathcal{S}$) computes the partition $\mathcal{S} = S_1 \cup S_2 \cup S_{12}$. Then, it recursively computes an approximately optimal solution for each of S_1 and S_2 and computes their disjoint union. This is one candidate approximate solution. Then, it computes an approximate solution to the instance with S_{12} as its input using Algorithm *IndSet*2, which is also a recursive procedure. This is another candidate approximate solution. *IndSet*1 then compares the two candidate solutions and outputs the one of larger size.

Now we give an outline of how Algorithm *IndSet*2 works. Note that the input to *IndSet*2 is a set S_{12} satisfying : for each $L \in S_{12}$, its horizontal arm intersects the vertical line $x = x_{med}$. We refer to this class of graphs formed by such sets (with every member intersecting a common vertical line) as vertical L-graphs (a formal definition is provided in the next chapter also).

This algorithm (on input T forming a vertical L-graph) is essentially Algorithm *IndSet*1 except that we use $L.cy$ and $L.vy$ values in place of $L.cx$ and $L.hx$ values to sort the L's, compute the median y_{med} and also for computing the partition $T = T_1 \cup T_2 \cup T_{12}$, in a way similar to how *IndSet*1 computes S_1, S_2, S_{12}. Precisely, the sets T_1, T_2, T_{12} are defined as follows.

$T_1 := \{L \in T \ : \ L.vy < y_{med}\}$.

$t_2 := \{L \in T \ : \ L.cy > y_{med}\}$.

$T_{12} := \{L \in T \ : \ L.cy \leqslant y_{med} \leqslant L.vy\}$.

The set T_{12} is a set satisfying : the horizontal and vertical arm of each member intersects a common vertical line $x = x_{med}$ and a common horizontal line $y = y_{med}$ respectively. It turns out (as established below in Lemma 16) that intersection graphs of such sets is a subclass of co-comparability graphs (complements of comparability graphs) and hence a MIS can be computed exactly and efficiently over such graphs (see [Gol04]). This is one candidate solution for $G[T]$. Approximate independent sets are computed recursively for each of the two sub-instances specified by T_1 and T_2 and their disjoint union is also computed which forms another candidate solution. As before, we compare the two candidate solutions and output the better one.

Algorithm 4 IndSet1

Require: A non-empty set S of L's.

1: **if** $|S| \leqslant 3$ **then**
2: **return** Compute and return a maximum independent set I_S of S
3: **else**
4: Compute x_{med} and also the partition $S = S_1 \cup S_2 \cup S_{12}$.
5: Compute $IndSet1(S_1) \cup IndSet1(S_2)$ and also $IndSet2(S_{12})$.
6: Return I_S defined as the larger of the two sets computed before.
7: **end if**

Algorithm 5 IndSet2

Require: A non-empty set T of Ls satisfying : for some vertical line $x = a$, each member of T intersects $x = a$.

1: **if** $|Y| \leqslant 3$ **then**
2: **return** Compute and return a maximum independent set I_Y of Y.
3: **else**
4: Compute y_{med} and also the partition $Y = Y_1 \cup Y_2 \cup Y_{12}$.
5: Compute $J_{union} = IndSet2(Y_1) \cup IndSet2(Y_2)$ and also
6: Compute a maximum independent set J^*_{12} of Y_{12}.
7: Return J_Y defined as the larger of the two sets computed before.
8: **end if**

The following lemma justifies how Step 6 of $IndSet2$ can be implemented efficiently.

Lemma 16. *Suppose S' is a set of L's. Suppose there exist a horizontal line $y = b$ and a vertical line $x = a$ such that each $L \in S'$ intersects both $y = b$ and $x = a$. Then, the intersection graph of members of S' is a co-comparability graph.*

Proof. We begin with the following claim.

Claim 2. *A pair $L_1, L_2 \in S'$ is independent if and only if $c_{L_1} < c_{L_2}$ or vice versa.*

Proof. (of Claim) It is easy to see that if either $c_{L_1} < c_{L_2}$ or $c_{L_2} < c_{L_1}$, then L_1 and L_2 are independent. To prove the converse : Assume that L_1 and L_2 are independent. By Claim 1, $L_1.cx \neq L_2.cx$ and $L_1.cy \neq L_2.cy$. Suppose that neither $c_{L_1} < c_{L_2}$ holds nor $c_{L_2} < c_{L_1}$ holds. As a consequence, we have one of the following two scenarios : (1) $L_1.cx < L_2.cx$ and $L_1.cy > L_2.cy$ **or** (2) $L_1.cx > L_2.cx$ and $L_1.cy < L_2.cy$. For Case (1), we have $(L_2.cx, L_1.cy) \in P_{L_1} \cap P_{L_2}$. For Case (2), we have $(L_1.cx, L_2.cy) \in P_{L_1} \cap P_{L_2}$. In both cases, we have used our assumption that both L_1 and L_2 intersect the lines $y = b$ and $x = a$. In either case, L_1 and L_2 intersect and hence are not independent, a contradiction to our assumption. □

Consider the complement of the intersection graph G formed by members of S'. Its vertices are members of S' and there is an edge between two members if and only if they do not intersect. We denote this graph by G^C. We orient each edge (L_1, L_2) as follows : it is oriented as $L_1\vec{,}L_2$ if $c_{L_1} < c_{L_2}$ and as $L_2\vec{,}L_1$ otherwise. Let $\vec{E}$ be the resulting orientation of E. Thus, to prove that G is a co-comparability graph, it suffices to show that $\forall L_i, L_j, L_k \in S'$, the following is true : $(L_i\vec{,}L_j \in \vec{E} \wedge L_j\vec{,}L_k \in \vec{E}) \Rightarrow (L_i\vec{,}L_k \in \vec{E})$. But by the above claim $L_i\vec{,}L_j \Rightarrow c_{L_i} < c_{L_j}$ and $L_j\vec{,}L_k \Rightarrow c_{L_j} < c_{L_k}$. It then follows that $c_{L_i} < c_{L_k}$. This implies that (L_i, L_k) is oriented as $L_i\vec{,}L_k$ by the above claim. This establishes the transitivity of the orientation and hence G is a co-comparability graph. This completes the proof of Lemma 16. □

4.3 Analysis of IndSet1 and IndSet2

Denote by I^* any maximum independent set of S. Similarly, denote by I_1^*, I_2^* and I_{12}^* any maximum independent set of S_1, S_2 and S_{12} respectively. Denote by I, I_1, I_2 and I_{12} the independent set produced by $IndSet1$ when provided with S, S_1, S_2 and S_{12} as input

respectively.

Lemma 17. $|I_{12}| \geqslant \frac{|I^*_{12}|}{\log |S_{12}|}$.

Proof. We use Y to denote the set S_{12}. Let $|Y| = m$. Let Y_1, Y_2, Y_{12} denote the partition of Y computed in Step 4 of $IndSet(S_{12})$. It follows that $|Y_1| \leqslant \frac{m}{2}$, $|Y_2| \leqslant \frac{m}{2}$ and $|Y_{12}| \leqslant \frac{m}{2}$ by our assumption stated in (i) of Claim 1. We prove the lemma by induction on m.

The base case is when $|Y| \leqslant 3$ or when $Y = Y_{12}$. For this case, we can solve the instance optimally since $|Y|$ is either small or its intersection graph is a co-comparability graph. This takes care of the base case.

Let J^*_1, J^*_2 and , J^*_{12} denote respectively a maximum independent set of Y_1, Y_2 and Y_{12}. Let J_1, J_2 and J_{12} denote respectively the solutions returned by $IndSet2$ when the input is Y_1, Y_2 and Y_{12}. Since Y_{12} induces a co-comparability intersection graph, we have $|J_{12}| = |J^*_{12}|$. Recall that I^*_{12} denote the maximum independent set of S_{12}. By induction, $|J_1| \geqslant \frac{|J^*_1|}{\log(m/2)} \geqslant \frac{|I^*_{12} \cap Y_1|}{\log m - 1}$, $|J_2| \geqslant \frac{|I^*_{12} \cap Y_2|}{\log m - 1}$. Thus,

$$\begin{aligned} I_{12} &= \max\{|J_{12}|, |J_1| + |J_2||\} \\ &\geqslant \max\{|I^*_{12} \cap Y_{12}|, \frac{|I^*_{12} \cap Y_1| + |I^*_{12} \cap Y_2|}{\log m - 1}\} \\ &\geqslant \max\{|I^*_{12} \cap Y_{12}|, \frac{|I^*_{12}| - |I^*_{12} \cap Y_{12}|}{\log m - 1}\} \end{aligned}$$

If $|I^*_{12} \cap Y_{12}| \geqslant \frac{|I^*_{12}|}{\log |S_{12}|}$ we are done. Otherwise,

$$\frac{|I^*_{12}| - |I^*_{12} \cap Y_{12}|}{\log m - 1} \geqslant \frac{|I^*_{12}| - |I^*_{12}|/\log m}{\log m - 1} = \frac{|I^*_{12}|}{\log |S_{12}|}.$$

This establishes the induction step, thereby completing the inductive proof. □

Recall that $|S| = n$.

Lemma 18. $I \geqslant \frac{|I^*|}{\log^2 n}$.

Proof. Due to our assumption stated in (i) of Claim 1, we have

$$|S_1| \leqslant \frac{n}{2}, \quad |S_2| \leqslant \frac{n}{2}, \quad |S_{12}| \leqslant \frac{n}{2}.$$

Again the proof is based on induction on n. We have the following.

$$I_1 \geqslant \frac{|I_1^*|}{\log^2(n/2)} \geqslant \frac{|I^* \cap S_1|}{(\log n - 1)^2} \tag{4.1}$$

$$I_2 \geqslant \frac{|I_2^*|}{\log^2(n/2)} \geqslant \frac{|I^* \cap S_2|}{(\log n - 1)^2} \tag{4.2}$$

From Lemma 17, we have

$$I_{12} \geqslant \frac{I_{12}^*}{\log |S_{12}|} \geqslant \frac{|I^* \cap S_{12}|}{\log |S_{12}|} \tag{4.3}$$

$$\begin{aligned} \text{Also, } |I| &= \max\{|I_{12}|, |I_1| + |I_2|\} \\ &\geqslant \max\{\frac{|I^* \cap S_{12}|}{\log |S_{12}|}, \frac{|I^*| - |I^* \cap S_{12}|}{(\log n - 1)^2}\} \end{aligned} \tag{4.4}$$

The last inequality follows from applying Inequalities (5.2), (5.3) and (5.3).

The base case corresponding to $n \leqslant 3$ follows since we can find a maximum independent set in constant time.

For an arbitrary $n > 3$, the inductive argument is as follows : If $\frac{|I^* \cap S_{12}|}{\log |S_{12}|} \geqslant \frac{|I^*|}{\log^2 n}$, the the induction step is proved. Otherwise, we have $|I^* \cap S_{12}| < \frac{|I^*| \log |S_{12}|}{\log^2 n}$. Thus,

$$\begin{aligned} \frac{|I^*| - |I^* \cap S_{12}|}{(\log n - 1)^2} &\geqslant \frac{|I^*| - \frac{|I^*| \log |S_{12}|}{\log^2 n}}{(\log n - 1)^2} \\ &\geqslant \frac{|I^*| - \frac{|I^*|}{\log n}}{(\log n - 1)^2} \\ &\geqslant \frac{|I^*|}{\log^2 n} \end{aligned}$$

This proves the induction step for the case when $\frac{|I^* \cap S_{12}|}{\log |S_{12}|} < \frac{|I^*|}{\log^2 n}$. Hence the proof. □

Lemma 18 establishes an upper bound of $(\log n)^2$ on the approximation factor of *IndSet*1 over L-graphs. By combining this observation with Lemma 15, one deduces that MIS over B_1-VPG graphs can be approximated efficiently within an approximation ratio of $4(\log n)^2$. We prove in the next subsection that *IndSet*1 runs in polynomial time. This leads us to the following theorem on approximating a maximum independent set over B_1-VPG graphs.

Theorem 5. *There exists polynomial time algorithm which, given a B_1-VPG graph $G = (\mathcal{S}, E)$ ($\mathcal{S}$ is a set of ℓ's), outputs an independent set of size at least $\frac{|I^*|}{4(\log n)^2}$ where I^* denotes any MIS of G and $n = |\mathcal{S}|$.*

4.3.1 Analysis of running time

Let $s(m)$ denote the running time of *IndSet*2(Y) on an input Y of size m. We have $s(m) = O(1)$ if $m \leqslant 3$. If Y induces a co-comparability graph, then $s(m) = O(m^2)$. Otherwise, $s(m) \leqslant 2s(m/2) + O(m^2)$. Unravelling the recursion, we deduce that $s(m) = O(m^2(\log m))$.

Let $t(n)$ denote the running time of *IndSet*1(S) on an input S of size n. We have $t(n) = O(1)$ if $n \leqslant 3$. Otherwise, $t(n) \leqslant 2t(n/2) + s(n/2) \leqslant 2t(n/2) + O(n^2(\log n))$. Unravelling the recursion, we deduce that $t(n) = O(n^2(\log n)^2)$. Thus, *IndSet*1(S) runs in time $O(n^2(\log n)^2)$ on an input of size n.

4.4 Approximation for equilateral B_1-VPG:

Maximum Independent Set in Equilateral B_1-VPG
Input: A set S of equilateral L's such that $\forall L \in S$.
Output: An independent set $I \subseteq S$ such that $\|I\|$ is maximized.

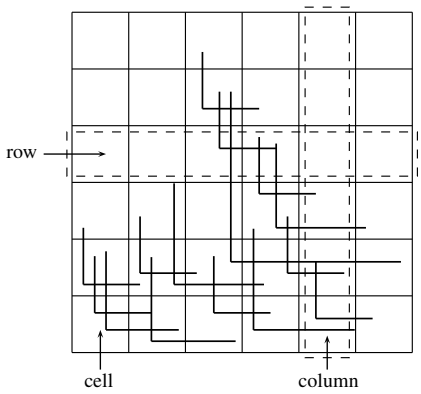


Figure 4.1: The grid is for L's of type 1 whose length varies within the range 2^i to 2^{i+1}

We call the above problem as *MISL*. We call an equilateral L a unit L if $le(L) = 1$. In Theorem 7, we establish that the decision version of MISL restricted to unit L's (and denoted by $MIS\,1$) is NP-Complete. As a consequence, it follows that the decision version of MISL is also NP-complete. In the rest of this section, we present a new approximation algorithm for MISL. Before that, we present a claim which can be justified easily. Let l_{min} be the minimum length of any arm in the given set of equilateral L's. Similarly, l_{max} denotes the maximum length of any arm.

Claim 3. *Without loss of generality, assume that the input to MISL satisfies* $l_{min} = 2$.

Proof. We rescale the coordinates of x-axis and y-axis by stretching both of them by a multiplicative factor of $2/l_{min}$. This makes $l_{min} = 2$. □

In view of Lemma 15 and the assumption of Claim 3, it suffices to focus only on equilateral L-graphs formed by a set S of equilateral L's where $l_{min}(S) = 2$. Define $d = l_{max}/l_{min}$.

The algorithm begins by dividing the input set S into disjoint sets $S_1, S_2, \ldots, S_{\lfloor \log 2d \rfloor}$ where $S_i = \{L \in S \mid 2^i \leqslant le(L) < 2^{i+1}\}$, $\forall i \in [\lfloor \log_2 2d \rfloor]$. This split is to exploit the fact that $l_{max}/l_{min} \leqslant 2$ when the input is restricted to only members of S_i, for any i. Using arguments similar to those employed in the proof of Lemma 15, one gets the following claim.

Lemma 19. *Suppose A is an efficient algorithm for solving MIS over the class of equilateral L-graphs satisfying* $\frac{l_{max}}{l_{min}} \leqslant 2$*, with an approxition ratio at most* $\alpha(n)$*. Then, there exists an efficient algorithm B which solves MIS over the class of equilateral L-graphs within a ratio of* $\alpha \cdot (\log_2 2d)$*, where* $d = \frac{l_{max}}{l_{min}}$ *for the input instance. It also follows that there exists an efficient algorithm C which solves MIS over the class of equilateral* B_1*-VPG graphs within a ratio of* $4\alpha \cdot (\log_2 2d)$*. For each of the algorithms, n stands for the size of the input.*

Thus, it suffices to describe how to obtain efficiently a good approximation of MIS for each i. Consider any fixed but arbitrary i and the corresponding $G_i = G[S_i]$. One proceeds as follows. We place a sufficiently large but finite grid structure on the plane covering all members of S_i. The grid is chosen in such a way so that grid-length in each of the x and y directions is 2^i. What we get is a rectangular array of square boxes of side length 2^i each. We number the rows of boxes from the bottom and the columns of boxes from left.

We label a *box* by (r', c') if it is in the intersection of r'^{th} row and c'^{th} column of boxes. We say L is *inside* a box if its corner c_l *either* lies in the interior of the box *or* lies on one of the left vertical boundary edge or the bottom horizontal boundary edge. If L lies inside a box (r', c') we denote it by $L \in (r', c')$.

We introduce some notations which will be used in subsequent discussions.

Consider a partition of S_i defined as follows : For every $k_r, k_c \in [3]$, define

$$S_{i,k_r,k_c} = \{L \in (r', c') \mid r' = k_r \mod 3, c' = k_c \mod 3\}.$$

Here, for purposes of simplicity, we use 3 in place of 0 in (mod 3) arithmetic. As an example $S_{i,1,1}$ consist of those Ls which belong to boxes indexed by
$\{(1,1),\ (1,4), ...,\ (4,1),\ (4,4), ...,\ (7,1),\ (7,7), ...\}$.

Thus, we partition input S_i into 9 subsets S_{i,k_r,k_c}. In Lemma 20, we establish that the

intersection graph $G[S_{i,1,1}]$ induced by $S_{i,1,1}$ is a co-comparability graph and hence, by symmetry, each of the 9 induced subgraphs is a co-comparability graph. Thus, for each of the 9 induced subgraphs, MIS can be solved exactly in polynomial time. We choose the largest of these 9 independent sets and return it as the output for $G[S_i]$. Assuming Lemma 20 (which we prove below) and combining all previous observations, we obtain the following Theorem 6.

Theorem 6. *There is an efficient* $36\lfloor \log 2d \rfloor$*-approximation for MIS over the class of* B_1*-VPG graphs. Here,* $d = l_{max}(S)/l_{min}(S)$ *is the ratio (defined before) associated with the instance.*

For proving Lemma 20 we introduce some notations. We consider the set $S_{i,1,1}$ and the complement of the corresponding intersection graph. We draw an edge between L_1, L_2 if L_1 and L_2 intersect. We denote this graph by $G(S_{i,1,1})$. Below we prove the following lemma.

Lemma 20. $G(S_{i,1,1})^C$ *is a comparability graph.*

Proof. Note that all members of $S_{i,1,1}$ lie in boxes which are in the intersection of rows and columns both numbered from $\{1, 4, 7, \ldots\}$. We prove the claim by showing that there exist a transitive orientation of the edges of this graph. We describe the orientation in two steps. First, we orient those edges which connect two L's whose corner lies in the same box. In the second step, we orient those edges which connect two L's located in two different boxes. For the first step, we employ the following claim which is an immediate consequence of Lemma 16.

Claim 4. *5 Suppose* L_1 *and* L_2 *are two members such that* $|L_1.cx - L_2.cx| \leqslant 2^i$ *and* $|L_1.cy - L_2.cy| \leqslant 2^i$*. Then,* L_1, L_2 *are independent if and only if* $c_{L_1} < c_{L_2}$ *or vice versa.*

Orientation : Let L_1 and L_2 be two arbitrary members of $S_{i,1,1}$ joined by an edge in $G(S_{i,1,1})^C$.

(*i*) If L_1 and L_2 are lying in a common box, we employ Claim 4 and orient it from L_1 to L_2 if $c_{L_1} < c_{L_2}$ and from L_2 to L_1 otherwise.

(*ii*) Suppose L_1 and L_2 lie in different boxes in the same row and let $L_1.cx < L_2.cx$ without loss of generality. We orient the edge from L_1 to L_2.

(*iii*) Suppose L_1 and L_2 lie in different rows and let $L_1.cy < L_2.cy$ without loss of generality. We orient the edge from L_1 to L_2.

If the orientation of an edge (L_1, L_2) is from L_1 to L_2, we denote it by $\overrightarrow{(L_1, L_2)}$.

We prove that this orientation is transitive. We prove it by performing a case analysis. For an edge $\overrightarrow{(L_1, L_2)}$, we call it *h-oriented* if vertices L_1 and L_2 lie in the same row and we call it *v-oriented* if L_1 lies in a row which is below the row in which L_2 is present. We denote by "*case h,v*", the case of 3 vertices L_1, L_2, L_3 such that (L_1, L_2) is h-oriented, (L_2, L_3) is v-oriented. Then we prove that there exist an edge $\overrightarrow{(L_1, L_3)}$. Similarly, the other cases "*h,h*", "*v,v*" and "*v,h*" are defined. We prove here the case "*h,h*". We handle the other cases similarly.

Case h, h: In this case we have three sub-cases. They are (1) L_1, L_2, L_3 are in the same box, (2) Two of the three vertices are in the same box different from the box of the other, (3) All three are in different boxes.

First, we handle the sub-case (1). L_1, L_2 are in the same box and they are independent. In view of Claim 4, this implies that $c_{L_1} < c_{L_2}$. Similarly, we infer that $c_{L_2} < c_{L_3}$. Hence, it follows that $c_{L_1} < c_{L_3}$ and hence (L_1, L_3) is oriented from L_1 to L_3. Thus it is transitive.

Now, for the sub-case (2) : either L_1, L_2 will be in the same box or L_2, L_3 will be in the same box. In both the cases L_1, L_3 will be in different boxes. Since any two points in different boxes of the same row differ in their x-coordinates by at least 2^{i+1}, the edge (L_1, L_3) exists and is directed L_1 to L_3, thereby proving the required transitivity.

The sub-case (3) : Since L's lie in different boxes in the same row, $L_3.cx - L_1.cx \geqslant 2^{i+2}$

and hence the edge (L_1, L_3) exists and is directed L_1 to L_3, thereby proving the required transitivity. This completes the proof of Case h, h.

Case h, v: Since L_1 and L_3 are in different rows, we have $L_3.cy - L_1.cy \geqslant 2^{i+1}$, the edge (L_1, L_3) exists and is directed L_1 to L_3, thereby proving the required transitivity.

Case v, h: In this case L_3 is in a box above that of L_1 by our hypothesis. As before, $L_3.cy - L_1.cy \geqslant 2^{i+1}$ and hence the edge (L_1, L_3) exists and is directed L_1 to L_3, thereby proving the required transitivity.

Case v, v: By our hypothesis L_3 is in a box above that of L_1. Hence, $L_3.cy - L_1.cy \geqslant 2^{i+2}$ and transitivity is established. □

4.5 Hardness of MIS on unit L-graphs

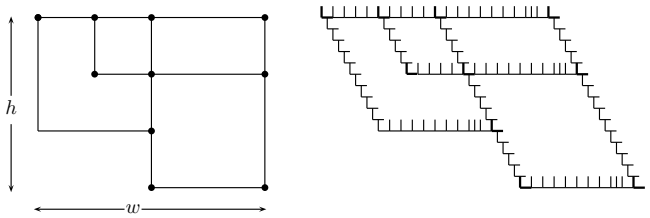


Figure 4.2: Planar graph with maximum degree four and its unit L VPG representation.

Theorem 7. *The decision version of* Maximum Independent Set *(MIS1) on unit L-graphs is NP-complete.*

Proof. Let $G = (V, E)$ be a planar graph with maximum degree four. It is known that Maximum Independent Set on a planar graph with maximum degree four is NP-complete [GJ79]. We construct an unit L-VPG representation of a planar graph with maximum degree four ($G' = (V', E')$) such that a maximum independent set in G' has a one to one correspondence to a maximum independent set of G, thereby proving our claim. Our proof is motivated by [KN90].

It is known that every planar graph of degree at most four can be drawn on a grid of linear size such that the vertices are mapped to points of the grid and the edges to piecewise linear curves made up of horizontal and vertical line segments whose endpoints are also points of the grid [Sch90]. It is reasonable to assume that a path between two vertices of G, if exists, use horizontal and vertical segments which have length more than one on the grid (otherwise it is possible to consider fine enough grid such that this property holds). Let $R(w, h)$ be the rectangular grid where the graph G has been drawn. We denote the width and height of the grid by w and h respectively. Let us consider $\delta = 1/2h$. Now for each vertex of the graph G, draw an unit length L whose corner point has co-ordinates $(x - \delta y, y)$, where in the grid $R(w, h)$ the vertex is positioned at (x, y). Let P_e be the path on the grid corresponding to edge e. Also let $|P_e|$ denote the number of intermediate grid vertices on the path P_e. Now for every path P_e, where $e = (u, v) \in E(G)$, if $|P_e|$ is even then for every intermediate grid vertex (x, y) on the path P_e draw a unit length L whose corner lie on $(x - \delta y, y)$. If $|P_e|$ is odd then for every intermediate grid vertex (x, y) except last one on the path P_e draw an unit length L whose corner lie on $(x - \delta y, y)$. If the last intermediate grid vertex (x, y) on the path is on a vertical segment of P_e then draw two L's as follows one L has corner point at $(x - \delta y, y - \epsilon)$ and other L has corner at $(x - \delta y, y)$ where $\epsilon > 0$ is a small number. If the last intermediate grid vertex (x, y) on the path is on a horizontal segment of P_e then draw two L's as follows one L has corner point at $(x - \delta y, y)$ and other L has corner at $(x - \delta y - \epsilon, y)$ where $\epsilon > 0$ is a small number. We denote this graph as G'. From the construction, it is clear that it is an intersection graph of unit L' s.

Clearly G' is obtained from G by subdividing every edge e with even number of new vertices (*even subdivision*). Let us denote the set of new vertices corresponding to an edge by V_e. Clearly $V' = V \cup_{e \in E(G)} V_e$.

Claim 5. *Let H denote a graph and H' be its even subdivision. There exists an independent set of H of size k if and only if there exists an independent set of size $k + \sum_{e \in E(H)} |V_e|/2$ in H'.*

Proof. Backward implication is easy to observe. Now we prove the other direction. Let us assume there exists an independent set I of $k + \sum_{e \in E(H)} |V_e|/2$ in H'. If $|I - I \cap V(H)| < \sum_{e \in E(H)} |V_e|/2$ remove all the subdivision vertices from the set. Otherwise $|I - I \cap V(H)| \geqslant \sum_{e \in E(H)} |V_e|/2$. Notice that $|V_e|$ is even for each of the edge $e \in H$. An independent set of H' contains at most half of the vertices of V_e. Hence $|I - I \cap V(H)| = \sum_{e \in E(H)} |V_e|/2$. Hence throw all the subdivision vertices as before. Hence the claim. □

Thus from the above claim we have $\alpha(G') = \alpha(G) + \sum_{e \in E(G)} |V_e|/2$. Thus we have exhibited a one to one relation between independent set of G' and independent set of G. Hence the proof. □

Chapter 5

Improved Approximation of MIS for B_1-VPG graphs

In this chapter, we present improved approximation algorithms for MIS over B_1-VPG graphs. In view of Lemma 15, it suffices to focus only on L-graphs. The algorithm is recursive and is essentially the one presented in Chapter 4 except that we design and employ a new exact algorithm for MIS over vertical L-graphs (that is, for $G[S_{12}]$). The previous algorithm of Chapter 4 employed a divide-and-conquer paradigm based recursive algorithm for this purpose. This exact and efficient algorithm for vertical L-graphs leads us to the improved approximation guarantee of $O(\log n)$ as against the previous one of $O((\log n)^2)$. Before we proceed further, we recall some definitions and assumptions.

5.0.1 Definitions and Assumptions

We state below some definitions and assumptions employed for the rest of this paper (employed in the previous chapter also).

Definition 4. *For a set S of (not necessarily distinct) real numbers, we define its median to be (i) the $\frac{n+1}{2}$-th smallest element if n is odd and (ii) the average of $\frac{n}{2}$-th and $(\frac{n}{2}+1)$-th*

smallest elements if n is even (*with ties being resolved arbitrarily or as explained in the specific application in sorting the numbers*).

Assumption (1) : Without loss of generality. the following holds throughout. If $\mathcal{L}$ is a set of L's, then $L_1.cx \neq L_2.cx$ and $L_1.cy \neq L_2.cy$, for any pair of distinct $L_1, L_2 \in \mathcal{L}$. That is, no two L's from $\mathcal{L}$ lie on the same vertical or horizontal line.

A formal justification of this Assumption (1) is provided at the end of this chapter.

5.1 $O(\log n)$-approximate algorithm for B_1-VPG graphs

As mentioned in the beginning of this chapter, we focus only L-graphs. We establish below that solving MIS approximately for L-graphs reduces to solving MIS exactly over vertical L-graphs which are defined below.

Definition 5. *A set $\mathcal{L}'$ of L-shaped objects is said to form a* vertical L-graph *if there exists a vertical line $x = a$ intersecting every $L \in \mathcal{L}'$.*

Outline: The broad outline of the improved algorithm is divide and conquer and is similar to the one employed in Chapter 4. We sort the objects in $\mathcal{L}$ in an increasing order of their cx values. Define x_{med} to be the median of the sorted values. Then, we compute the sets S_1, S_2 and S_{12} defined as follows.

$$S_1 := \{L \in \mathcal{L} \ : \ L.hx < x_{med}\}.$$

$$S_2 := \{L \in \mathcal{L} \ : \ L.cx > x_{med}\}.$$

$$S_{12} := \{L \in \mathcal{L} \ : \ L.cx \leqslant x_{med} \leqslant L.hx\}.$$

The sets S_1, S_2 and S_{12} form a partition of $\mathcal{L}$. Also, any pair of $L_1 \in S_1, L_2 \in S_2$ are independent. In addition, members of S_{12} induce a vertical L-graph. The problem is solved

by applying the recursive Algorithm $IndSet1$. $IndSet3(\mathcal{L})$ is an exact algorithm for MIS applied when $\mathcal{L}$ induces a vertical L-graph. This algorithm (on input $\mathcal{L}$) computes the partition $\mathcal{L} = S_1 \cup S_2 \cup S_{12}$. Then, it recursively computes an approximately optimal solution for each of S_1 and S_2 and computes their disjoint union. This is one candidate approximate solution. Then, $IndSet3$ computes exactly a MIS of $G[S_{12}]$. This is another candidate approximate solution. $IndSet1$ then compares the two candidate solutions and outputs the one of larger size. The following theorem establishes that designing an effi-

Algorithm 6 IndSet1

Require: A non-empty set $\mathcal{L}$ of L's.

1: **if** $|L| \leqslant 3$ **then**
2: **return** Compute and return a maximum independent set $I(\mathcal{L})$ of $\mathcal{L}$
3: **else**
4: Compute x_{med} and also the partition $\mathcal{L} = S_1 \cup S_2 \cup S_{12}$.
5: Compute $IndSet1(S_1) \cup IndSet1(S_2)$ and also $IndSet3(S_{12})$.
6: Return $I(\mathcal{L})$ defined as the larger of the two sets computed before.
7: **end if**

cient, $\alpha(n)$-approximate algorithm for vertical L-graphs leads to the design of an efficient, $\alpha(n)(\log n)$-approximate algorithm for L-graphs. In what follows, we use $I^*(S)$ to denote a MIS of the graph induced by S.

Theorem 8. *Let $\alpha(n)$ be an arbitrary non-decreasing function of n. Suppose $IndSet3$ is an an efficient, $\alpha(n)$-approximate MIS algorithm over vertical L-graphs. Then, $IndSet1$ is an efficient, $\alpha(n)(\log n)$-approximate MIS algorithm over L-graphs. For both approximation algorithms, n stands for the size of the input.*

Proof. We have the following : $|S_1| \leqslant \frac{n}{2}$, $|S_2| \leqslant \frac{n}{2}$. We prove the above claim using induction on n. For the base case of $n \leqslant 3$, we can obtain a MIS in constant time. Now consider the case when $n > 3$. Let $I_1 = IndSet1(S_1)$, $I_2 = IndSet1(S_2)$ and $I_{12} =$

$IndSet2(S_{12})$. Let $I_1^* = I^*(S_1)$, $I_2^* = I^*(S_2)$ and $I_{12}^* = I^*(S_{12})$. By induction hypothesis,

$$|I_1| \geqslant \frac{|I_1^*|}{\alpha(n/2)\log(n/2)} \geqslant \frac{|I^* \cap S_1|}{\alpha(n/2)\log(n/2)}, \tag{5.1}$$

$$|I_2| \geqslant \frac{|I_2^*|}{\alpha(n/2)\log(n/2)} \geqslant \frac{|I^* \cap S_2|}{\alpha(n/2)\log(n/2)}, \tag{5.2}$$

$$|I_{12}| \geqslant \frac{|I^* \cap S_{12}|}{\alpha(n)}. \tag{5.3}$$

Thus, $IndSet1(\mathcal{L})$ outputs a solution I satisfying

$$\begin{aligned} |I| &= \max\{|I_{12}|, |I_1| + |I_2|\} \\ &\geqslant \max\{\frac{|I^* \cap S_{12}|}{\alpha(n)}, \frac{|I^* \cap S_1| + |I^* \cap S_2|}{\alpha(n/2)\log(n/2)}\} \\ &= \max\{\frac{|I^* \cap S_{12}|}{\alpha(n)}, \frac{|I^*| - |I^* \cap S_{12}|}{\alpha(n/2)\log(n/2)}\}. \end{aligned} \tag{5.4}$$

If $\frac{|I^* \cap S_{12}|}{\alpha(n)} \geqslant \frac{|I^*|}{\alpha(n)\log n}$, we are done. Otherwise, we have

$$|I^* \cap S_{12}| \leqslant \frac{|I^*|}{\log n} \tag{5.5}$$

It follows from Inequalities (5.4) and (5.5) that

$$\frac{|I^*| - |I^* \cap S_{12}|}{\alpha(n/2)\log(n/2)} \geqslant \frac{|I^*| - \frac{|I^*|}{\log n}}{\alpha(n/2)\log(n/2)} = \frac{|I^*|}{\alpha(n/2)\log n} \geqslant \frac{|I^*|}{\alpha(n)\log n} \tag{5.6}$$

The last inequality follows since $\alpha(n)$ is a non-decreasing function. □

In the next section, we present an efficient and exact algorithm for finding a MIS in vertical L-graphs. As a consequence, we obtain the following conclusion.

Theorem 9. *$IndSet1$ is an efficient, $(\log n)$-approximate algorithm for MIS on L-graphs. As a consequence, one gets an efficient $4(\log n)$-approximate MIS algorithm over B_1-VPG graphs.*

Proof. Follows from Theorem 8 (by setting $\alpha(n) = 1$ for every n), since (as is shown in the following subsection) MIS on vertical L-graphs can be solved exactly in polynomial time. □

5.1.1 An exact algorithm for MIS on vertical L-graphs

Let $\mathcal{S}$ be a set of L's inducing a vertical L-graph G. We present an exact algorithm for finding a MIS in G. The algorithm is recursive and efficiency is achieved by implementing it using the Dynamic Programming paradigm. It involves computing a MIS in each of a polynomial number of smaller subproblems to get a MIS for the given input. The main intuition behind the efficiency is an appropriate formulation of the recursion which helps us to bound the number of subproblems that need to be solved eventually.

We assume that each subproblem S comes equipped with two L's one on the top of all members of S (and referred to as a cap) and the other one (referred to as a cushion) is to the left and bottom of all members of S. Both cap and cushion are not members of S. There are two advantages in introducing cap and cushion: it provides a brief and concise description of the subproblems, it also helps to obtain a simple derivation of the polynomial bound on the number of subproblems. The two notions and some others are introduced below. They play a very useful role in obtaining a concise description of the recursive computation of optimal solutions.

Definition 6. *Let L, L' be two arbitrary L's. We say that $L <_x L'$ if $L.cx < L'.cx$. We say that $L <_y L'$ if $L.cy > L'.cy$.*

Definition 7. *Let L, L' be two arbitrary L's. We say that L' is* entirely right and below *of L if (i) $L <_x L'$, (ii) $L <_y L'$ and (iii) $L'.vy < L.cy$. We say that L' is* entirely right and above *of L if $c_L < c_{L'}$.*

Definition 8. *Let $\mathcal{S}$ be an arbitrary set of Ls such that each member intersects a common vertical line $x = a$. A* (cap, cushion) *of $\mathcal{S}$ is any pair (L_1, L_2) of L's each intersecting $x = a$*

such that (i) *each* $L' \in \mathcal{S}$ *is entirely right and below of* L_1, (ii) *each* $L' \in \mathcal{S}$ *is entirely right and above of* L_2, (iii) L_2 *is entirely right and below of* L_1.

Definition 9. *Let* $\mathcal{S}$ *be an arbitrary set of Ls such that each member intersects a common vertical line* $x = a$. *Let* (L_1, L_2) *be a pair of L's also intersecting* $x = a$ *such that* L_2 *is entirely right and below of* L_1. *We define the subset of* $\mathcal{S}$ *capped and cushioned by* (L_1, L_2) *to be the set of those* $L \in \mathcal{S}$ *such that* (i) L *is entirely right and below* L_1 *and* (ii) L *is entirely right and above* L_2. *We denote this set by* $\mathcal{S}_{L_1,L_2}$.

Definition 10. *Given a* $\mathcal{S}$ *with a cap* L *and a* $L'' \in \mathcal{S} \cup \{L\}$, *we use* $\mathcal{S}^{L''}$ *to denote the subset of those* $L' \in \mathcal{S}$ *which are smaller or equal to* L'' *with respect to* $<_y$ *ordering, that is, the set* $\{L' \in \mathcal{S} : L' <_y L'' \vee L' = L''\}$. *In particular, we have* $\mathcal{S} = \mathcal{S}^{L_s}$ *always where* L_s *is the last element of* $\mathcal{S}$ *with respect to* $<_y$ *ordering. Also,* $\mathcal{S}^{L} = \varnothing$ *always.*

Definition 11. *For a set* $\mathcal{S}$ *capped and cushioned by* (L, L') *with* L_s *being the last element (with respect to* $<_y$ *ordering), let* $LA(\mathcal{S}, L_s)$ *denote the set of those* $L'' \in \mathcal{S} \cup \{L\}$ *such that either* (i) $L'' = L$ *or* (ii) $L'' \in \mathcal{S} \setminus \{L_s\}$ *and* L_s *is entirely right and below* L''.

Definition 12. *For a set* $\mathcal{S}$ *inducing a vertical L-graph G, capped and cushioned by* (L, L'), *we use* $Opt(\mathcal{S}, L, L')$ *to denote any MIS in G.*

Definition 13. *For a finite sequence* $(A_1, \ldots, A_n)$ *of finite sets, let* $\max\{A_1, \ldots, A_m\}$ *denote the first set of maximum size in the sequence.*

Our algorithm is recursive and is based on the following recursion satisfied by $Opt(\mathcal{S}, L, L')$. The proof of the following lemma is provided in the appendix.

Lemma 21. *Let* $\mathcal{S}, L, L'$ *be as in the previous definition with* L_s *being the last memebr of* $\mathcal{S}$ *with respect to* $<_y$ *ordering. Then,* $Opt(\mathcal{S}, L, L')$ *equals (in size)*

$$\max\left\{Opt(\mathcal{S} \setminus \{L_s\}, L, L'),\ \max_{L'' \in LA(\mathcal{S},L_s)} \left\{ \{L_s\} \cup Opt(\mathcal{S}^{L''}, L, L') \ \cup\ Opt(\mathcal{S}_{L'',L_s}, L'', L_s) \right\}\right\},$$

provided $|\mathcal{S}| \geqslant 2$. *Otherwise,* $Opt(\mathcal{S}, L, L') = \mathcal{S}$.

Proof. For each $L'' \in LA(\mathcal{S}, L_s)$ (in the recursion given above), the set corresponding to L'' (in the max{.} expression) is an independent set in $G[\mathcal{S}]$. Let I^* be a fixed but arbitrary MIS in $G[\mathcal{S}]$. Consider the following three cases.

Case 1 $L_s \notin I^*$. Then, it should be the case that I^* is a MIS in $G[\mathcal{S} \setminus \{L_s\}]$. Also, (L, L') continue to cap-cushion $\mathcal{S} \setminus \{L_s\}$. Hence $I^* = Opt(\mathcal{S} \setminus \{L_s\}, L, L')$.

Case 2 $L_s \in I^*$ and $LA(\mathcal{S}, L_s) \cap I^* = \emptyset$. When $L'' = L$, $\mathcal{S}^{L''} = \emptyset$ and it should also be that $I^* \setminus L_s$ is a MIS in $G[\mathcal{S}_{L,L_s}]$ where $\mathcal{S}_{L,L_s}$ is capped and cushioned by (L, L_s).

Case 3 $L_s \in I^*$ and $LA(\mathcal{S}, L_s) \cap I^* \neq \emptyset$. Then, it should be that $LA(\mathcal{S}, L_s) \neq \{L\}$. Let L'' be the last member of $LA(\mathcal{S}, L_s) \cap I^*$. In that case, I^* is the disjoint union of $\{L_s\}$, $I^* \cap \mathcal{S}_{L'',L_s}$ and $I^* \cap \mathcal{S}^{L''}$. Also, $I^* \cap \mathcal{S}_{L'',L_s}$ should be a MIS in $G[\mathcal{S}_{L'',L_s}]$ with (L'', L_s) as its cap-cushion. Similarly, $I^* \cap \mathcal{S}^{L''}$ should be a MIS in $G[\mathcal{S}^{L''}]$ with (L, L') as its cap-cushion.

This completes the proof. □

Suppose $\mathcal{S}$ is a set of n members inducing a vertical L-graph with $x = a$ being the common vertical line. Let $(L_1, L_2, \ldots, L_n)$ be the linear ordering of $\mathcal{S}$ defined by $L_i <_y L_j$ for every $i < j$. Choose a cap and cushion (L_0, L_{n+1}) for $\mathcal{S}$. It is easy to see that one can always compute such a pair in linear time. The correctness and the claim of polynomial time bound are based on the following series of claims whose proofs are provided in the appendix. Let T denote the unique recursion tree capturing the recursion based computation of $Opt(\mathcal{S}, L_0, L_{n+1})$.

Claim 6. *The problem size ($|\mathcal{S}|$) keeps decreasing along every path in T until it reaches the base case $|\mathcal{S}| \leqslant 1$.*

Proof. Each of the sets $\mathcal{S} \setminus \{L_s\}, \mathcal{S}^{L''}, \mathcal{S}_{L'',L_s}$ has a size which is less than that of $\mathcal{S}$. □

Claim 7. *Each of the sets $\mathcal{S}$ defining a subproblem is a subset of the original input $\{L_1, \ldots, L_n\}$.*

Proof. The proof is based on the depth of recursion from the root of T. The claim is obviously true for the root. Each of the sets $\mathcal{S} \setminus \{L_s\}, \mathcal{S}^{L''}, \mathcal{S}_{L'',L_s}$ is a subset of $\mathcal{S}$ which is the input for the current subproblem. □

Claim 8. *Every pair (L, L') of (cap,cushion) that arises in any subproblem generated by the above recursion is of the form (L_i, L_j) where $0 \leqslant i < j \leqslant n+1$.*

Claim 9. *Each of the sets $\mathcal{S}'$ defining a subproblem is a set of the form $\mathcal{S}_{L_i,L_j}^{L_k}$ for some $0 \leqslant i \leqslant k < j \leqslant n+1$ and $\mathcal{S} = \{L_1, \ldots, L_n\}$.*

As a consequence, we obtain the following corollary.

Claim 10. *There are at most n^3 distinct subproblems that are actually solved in the recursion formulation.*

Continuing further, we obtain the following theorem.

Theorem 10. *There exists an $O(n^4)$ time exact algorithm for finding a MIS in vertical L-graphs.*

Proof. We employ the Dynamic Programming by first enumerating all possible subproblems and then find solutions to these subproblems in a bottom-up approach starting with the base cases. Computation of the sets $\mathcal{S}_{L_i,L_j}^{L_k}$ and finding the sizes of optimum solutions can be combined to yield an $O(n^4)$ time algorithm for solving MIS in vertical L-graphs. □

5.2 Appendix 1: Proof of Assumption (1) :

Two sets $\mathcal{L}, \mathcal{L}''$ of L's are said to be *equivalent* if $G[\mathcal{L}]$ and $G[\mathcal{L}'']$ are isomorphic. The proof of Assumption (1) is achieved in two steps. First, given a set $\mathcal{L} = \{L_1, \ldots, L_n\}$, we prove that there exists an efficiently computable and equivalent (to $\mathcal{L}$) $\mathcal{L}' = \{L'_1, \ldots, L'_n\}$

such that (*i*) $L'_i.cy = L_i.cy$ for each i and (*ii*) $L'_i.cx \neq L'_j.cx$ for every $i \neq j$. By symmetrical arguments, it also follows that there exists an efficiently computable and equivalent (to $\mathcal{L}'$) $\mathcal{L}'' = \{L''_1, \ldots, L''_n\}$ such that (*i*) $L''_i.cx = L'_i.cx$ for each i and (*ii*) $L''_i.cy \neq L''_j.cy$ for every $i \neq j$. $\mathcal{L}''$ is the required set of L's. By symmetry, it suffices to prove only the existence and efficient computation of $\mathcal{L}'$. Existence and computation of $\mathcal{L}''$ (from $\mathcal{L}'$) is similar. Existence and efficient computation of $\mathcal{L}'$ follows from the following two claims.

Claim 11. *For every finite set $\mathcal{L} = \{L_i\}_i$ of L's, there exists an efficiently computable and equivalent $\mathcal{L}^a = \{L^a_i\}_i$ such that (A) $L^a_i.hx \neq L^a_j.cx$ for every $i \neq j$, (B) $L^a_i.cy = L_i.cy$ for every i and (C) vertical and horizontal arms of L^a_i have the same respective lengths as those of L_i, for every i.*

Proof. Let $x_1 < \ldots < x_m < \infty = x_{m+1}$ be the sorted list of m distinct reals (after ignoring multiple occurrences) that appear in $\{L_i.cx\}_i$. Define, for $k \geqslant 2$, $\alpha_k = x_k - x_{k-1}$ and $\alpha = \min\{\alpha_k \ : \ k \geqslant 2\}$. For every i, define β_i as follows : If $x_k < L_i.hx < x_{k+1}$ for some $k \leqslant m$, then define β_i to be $\min\{L_i.hx - x_k, x_{k+1} - L_i.hx\}$. If $L_i.hx = x_k$ for some $2 \leqslant k \leqslant m$, define β_i to be α_k. Let β be the minimum of β_i over all i. Define γ to be $\min\{\frac{\alpha}{2n}, \frac{\beta}{2n}\}$ where $n = |\mathcal{L}|$. We have $\gamma > 0$.

For every i, define L^a_i as below : Let $L_i.cx = x_k$. L^a_i is the same as L_i (with exactly same length vertical and horizontal arms) except that its corner point is shifted in the negative x direction by $k\gamma$ distance. That is, L^a_i is characterized by $(x_k - k\gamma, L_i.cy, L_i.hx - k\gamma, L_i.vy)$.

That $\mathcal{L}^a$ satisfies condition (A) can be established as follows : To have $L^a_i.hx = L^a_j.cx$ for some $i \neq j$, we should have $L_i.cx = x_r < x_t = L_j.cx$ for some $r < t$. Otherwise, $L^a_j.cx \leqslant L_j.cx = x_t \leqslant x_r < L^a_i.hx$. Suppose $L_i.hx < x_t$. Then, $L^a_i.hx \leqslant L_i.hx < L^a_j.cx$. If $L_i.hx > x_t$, then, $L^a_j.cx \leqslant x_t < L^a_i.hx$. If $L_i.hx = L_j.cx$, then, $L^a_j.cx = x_t - t\gamma < x_t - r\gamma = L^a_i.hx$.

We establish that $\mathcal{L}^a$ and $\mathcal{L}$ are equivalent by establishing that for every $i \neq j$, L^a_i and L^a_j are independent if and only if L_i and L_j are independent. Fix an arbitrary $i \neq j$. Without loss of generality, assume that $L_i.cx \leqslant L_j.cx$.

Suppose, $L_i.cx = L_j.cx = x_k$. Then, $L_i^a.cx = L_j^a.cx = x_k - k\gamma$. Clearly, L_i^a and L_j^a intersect if and only if L_i and L_j intersect. Hence, from now onwards, assume that $L_i.cx < L_j.cx$. Let $L_j.cx = x_k$ where $k \geqslant 2$.

If $L_i.hx < L_j.cx$, then L_i and L_j are independent. Also, $L_j^a.cx = L_j.cx - k\gamma \geqslant L_j.cx - \frac{\beta}{2} > L_i.hx \geqslant L_i^a.hx$ and hence L_i^a and L_j^a are independent.

Suppose we have $L_j.cx \leqslant L_i.hx$. We have $L_j^a.cx = x_k - k\gamma \geqslant x_k - \frac{\alpha}{2} > x_{k-1} \geqslant L_i.cx \geqslant L_i^a.cx$. If $L_j.cx < L_i.hx$, then $L_i^a.hx \geqslant L_i.hx - k\gamma \geqslant L_i.hx - \frac{\beta}{2} > L_j.cx > L_j^a.cx$. If $L_j.cx = L_i.hx$, then $L_i^a.hx \geqslant x_k - (k-1)\gamma > x_k - k\gamma = L_j^a.cx$. In any case, we have $L_i^a.cx < L_j^a.cx < L_i^a.hx$. Hence, L_i^a and L_j^a intersect if and only if $L_j^a.cy \leqslant L_i^a.cy \leqslant L_j^a.vy$. Similarly, L_i and L_j intersect if and only if $L_j.cy \leqslant L_i.cy \leqslant L_j.vy$. In other words, L_i^a and L_j^a intersect if and only if L_i and L_j intersect. □

Claim 12. *For every $\mathcal{L}^a$ mentioned before, there exists an efficiently computable and equivalent $\mathcal{L}' = \{L_i'\}_i$ such that (D) $L_i'.cx \neq L_j'.cx$ for every $i \neq j$, (E) $L_i'.cy = L_i^a.cy$ for every i and (F) vertical and horizontal arms of L_i^a have the same respective lengths as those of L_i' for every i.*

Proof. Let $(x_k)_k$, $(\alpha_k)_k$, $(\beta_i)_i$, α and β be defined as in the proof of Claim 11 with $\mathcal{L} = \mathcal{L}^a$. For every i, define δ_i to be $L_i^a.hx - L_i^a.cx$ and define δ to be $\min_i \delta_i$. Redefine γ to be $\min\{\frac{\alpha}{2n}, \frac{\beta}{2n}, \frac{\delta}{2n}\}$ where $n = |\mathcal{L}|$. We have $\gamma > 0$.

Define $\mathcal{L}'$ in terms of $\mathcal{L}^a$ as follows. Fix an arbitrary k and order all those L's in $\mathcal{L}^a$ having $L.cx = x_k$ as follows. If L_1^a, L_2^a are two such members, then $L_1^a < L_2^a$ (or vice versa) if $L_1^a.cy > L_2^a.cy$. If $L_1^a.cy = L_2^a.cy$, then $L_1^a < L_2^a$ (or vice versa) if $L_1^a.hx < L_2^a.hx$. If $L_1^a.cy = L_2^a.cy$ and $L_1^a.hx = L_2^a.hx$, then $L_1^a < L_2^a$ (or vice versa) if $L_1^a.vy < L_2^a.vy$. Surely, any pair of distinct L's differ in at least one of the four values. Let $(L_1^a, L_2^a, \ldots, L_s^a)$ be the resulting total ordering. For every $i \leqslant s$, define L_i' as the L characterized by $(L_i^a.cx + (i-1)\gamma, L_i^a.cy, L_i^a.hx + (i-1)\gamma, L_i^a.vy)$. Note that $L_i'.cx - L_i^a.cx \leqslant (n-1)\gamma \leqslant \min\{\frac{\alpha}{2}, \frac{\beta}{2}, \frac{\delta}{2}\}$ for every i. Similarly, $L_i'.hx - L_i^a.hx \leqslant \min\{\frac{\alpha}{2}, \frac{\beta}{2}, \frac{\delta}{2}\}$ for every i.

That Condition (D) is satisfied by $\mathcal{L}'$ can be seen as follows. Fix an arbitrary $i \neq j$ satisfying $L_i^a.cx = x_r \leqslant x_t = L_j^a.cx$. If $x_r < x_t$, then $L_i'.cx \leqslant x_r + \frac{\alpha}{2} < x_t \leqslant L_j'.cx$. If $x_r = x_t$, let $L_i^a < L_j^a$ (without loss of generality) with respect to the ordering associated with x_r. Then, it follows from the definition that $L_i'.cx < L_j'.cx$.

We only need to show that for every $i \neq j$, L_i' and L_j' are independent if and only if L_i^a and L_j^a are independent. Fix an arbitrary $i \neq j$. Without loss of generality, assume that $L_i^a.cx \leqslant L_j^a.cx$.

Suppose $L_i^a.cx = L_j^a.cx = x_k$ with $L_i^a < L_j^a$ (without loss of generality) with respect to the ordering associated with x_k. There are two sub-cases.

(i) Suppose $L_i^a.cy > L_j^a.cy$. Then, L_i^a and L_j^a intersect if and only if $L_j^a.cy \leqslant L_i^a.cy \leqslant L_j^a.vy$. Also, L_i' and L_j' intersect if and only if $L_j'.cy = L_j^a.cy \leqslant L_i'.cy = L_i^a.cy \leqslant L_j'.vy = L_j^a.vy$, since $L_i'.cx \leqslant L_j'.cx \leqslant x_k + \frac{\delta}{2} < L_i^a.hx \leqslant L_i'.hx$. Thus, L_i^a and L_j^a intersect if and only if L_i' and L_j' intersect.

(ii) Suppose $L_i^a.cx = L_j^a.cx = x_k$ and $L_i^a.cy = L_j^a.cy$. Then, L_i^a and L_j^a intersect. Also, $L_i'.cx \leqslant L_j'.cx < L_i'.hx$. Hence, L_i' and L_j' also intersect.

Hence, from now on, assume that $L_i^a.cx = x_r < x_t = L_j^a.cx$. If $L_i^a.hx < L_j^a.cx$, then L_i^a and L_j^a are independent. Also, $L_i'.hx \leqslant L_i^a.hx + \frac{\beta}{2} < L_j^a.cx \leqslant L_j'.cx$. Hence, L_i' and L_j' are independent.

The only remaining case (follows from Assumption (A) satisfied by $\mathcal{L}^a$) is that $L_i^a.cx = x_r < x_t = L_j^a.cx < L_i^a.hx$. Then, L_i^a and L_j^a intersect if and only if $L_j^a.cy = L_j'.cy \leqslant L_i^a.cy = L_i'.cy \leqslant L_j^a.vy = L_j'.vy$. Also, $L_i'.cx \leqslant x_r + \frac{\alpha}{2} < x_t \leqslant L_j'.cx \leqslant x_t + \frac{\beta}{2} < L_i^a.hx \leqslant L_i'.hx$. Hence, L_i^a and L_j^a intersect if and only if L_i' and L_j' intersect. This establishes that $\mathcal{L}^a$ is equivalent to $\mathcal{L}'$. □

Chapter 6

Approximation of MIS for B_2-VPG graphs

In this chapter, we present approximation algorithms for MIS over B_2-VPG graphs. Recall (from Section 1.2) that each member of a set defining a B_2-VPG graph is one of only sixteen possible shapes with exactly 2 bends. In particular, recall that two specific shapes of these are referred to as Z- and U-shapes. We refer to the class of graphs formed by a collection of Z-shapes as Z-graphs. The class of U-graphs is similarly defined. Similarly, one can define the class of vertical U graphs as the class of graphs formed by collections of U-shapes each of which intersects a common vertical line $x = a$. The class of vertical Z-graphs is similarly defined.

As we will see later, it suffices to focus only on designing efficient approximation algorithms for U-graphs and Z-graphs. Each of these algorithms is recursive and is similar to the one presented in Chapter 5 (with appropriate changes). It calls for the design of approximate algorithms for MIS over each of the vertical U-graphs, Z- graphs. We design a 2-approximate algorithm for vertical U-graphs and also design a $2(\log n)$-approximate algorithm for vertical Z-graphs. These efficient, approximate algorithms for vertical U-, Z-graphs lead us to efficient algorithms respectively for U-, Z-graphs, with respective

approximation guarantees of $2(\log n)$ and $2(\log n)^2$. Before we proceed further, we recall some definitions and assumptions.

Recall the definition of the median introduced in the previous chapters. The following assumption is used for the rest of this chapter. A formal justification of this Assumption (2) is provided at the end of this chapter.

Assumption (2) : Without loss of generality. the following holds throughout. If $\mathcal{U}$ is a set of U's, then $U_1.cx \neq U_2.cx$ and $U_1.cy \neq U_2.cy$, for any pair of distinct $U_1, U_2 \in \mathcal{U}$. That is, no two U's from $\mathcal{U}$ lie on the same horizontal line. Also, the left vertical arms of no two U's lie on the same vertical line.

6.1 Preliminaries

B_2-VPG graphs are formed by the following 8 shapes shown in the figure. The first four of these shapes are equivalent in the sense that one can obtain any of these shapes from any other by either or both of a 90^o-rotation and a reflection about X- and Y- axes, as the case demands. Similarly, the last four are equivalent. For ease of description, we intentionally refer to the fourth of the first four shapes as a Z shape. We refer to the last of the last four shapes as a $\sqcup$ shape. We focus only on Z and $\sqcup$ shapes. Other six shapes are treated similarly, in view of the symmetries between them. Both of these shapes has only one horizontal arm and two vertical arms, one on the left side and the other on right side. The intersection of the horizontal arms with the two vertical arms are called the corner points and the left corner point is denoted by c_1 and the right corner point is denoted by c_2. We use c_1x, c_1y, c_2x, c_2y to denote respectively the x, y-coordinates of the corner points c_1 and c_2. The y-coordinates of the tips of the left and right vertical arms of a $\sqcup$ are denoted respectively by ly and ry. The y-coordinates of the tips of the top and bottom vertical arms of a Z shape are denoted respectively by ty and dy. Any $\sqcup$ object is completely characterized by the six-tuple $(c_1x, c_1y, c_2x, c_2y, ly, ry)$ and any Z object is

completely characterized by $(c_1x, c_1y, c_2x, c_2y, ty, dy)$.

Depending on the shape of objects, we introduce new classes of graphs.

Definition 14. *A Z-graph is the intersection graph of Z-shaped geometric objects in the plane.*

Definition 15. *A U-graph is the intersection graph of ⊔-shaped geometric objects in the plane.*

6.2 Approximation algorithms for B_2-VPG graphs

Using the symmetries between ⊔- and Z-shaped objects and the other six objects (as described in Section 6.1), one can obtain the following analogue of Lemma 15 for B_2-VPG graphs.

Lemma 22. *If A and B are two efficient algorithms for solving MIS approximately over U-graphs and Z-graphs respectively, each with a performance guarantee bounded by $\alpha(n)$, then there exists an efficient algorithm C for solving MIS over B_2-VPG graphs, with a performance guarantee at most $8\alpha(n)$. Here, n stands for the size of the input for both algorithms.*

Proof. The symmetries between U- and Z-shaped objects implies that for each of the eight shapes that a path with two bends can take, MIS can be efficiently approximated within a multiplicative factor of $\alpha(n)$ for the class of intersection graphs induced by objects of that specific shape. By applying an appropriate approximation algorithm over each of the eight induced subgraphs and choosing the best of the eight solutions, one can solve MIS for any B_2-VPG graph within a factor of $8\alpha(n)$. □

In Section 6.3, it is shown that MIS can be efficiently approximated over U-graphs within a multiplicative factor of $2(\log n)$. In Section 6.4, it is shown that MIS can be efficiently

approximated over Z-graphs within a multiplicative factor of $2(\log n)^2$. Now, an application of Lemma 22 leads us to the following conclusion.

Theorem 11. *There exists an efficient,* $16(\log n)^2$*-approximate algorithm for MIS over the class of* B_2*-VPG graphs.*

6.3 Approximation algorithm for U-graphs

We use the phrase U-graphs to denote the class of all U-graphs. A subclass of U-graphs is the vertical U-graphs.

Definition 16. *A set* $\mathcal{U}$ *of* $\sqcup$*-shaped objects is said to form a* vertical U-graph *if there exists a vertical line* $x =$ *a intersecting every* $U \in \mathcal{U}$*.*

The approximate MIS algorithm *IndSet*1 designed for L-graphs also works for U-graphs. The reason is a $\sqcup$-shape is the same as a L-shape except for the vertical arm added at the tip of the horizontal arm of a L. The algorithm is the same. As for computing the median and partitioning, the $U.c_1x$ and $U.c_2x$ values take respectively the roles of $L.cx$ and $L.hx$ values. As before, we compute the median and partition $\mathcal{U}$ into $\mathcal{U} = \mathcal{S}_1 \uplus \mathcal{S}_2 \uplus \mathcal{S}_{12}$ where

$$\mathcal{S}_1 := \{U \in \mathcal{U} \ : \ U.c_2x < x_{med}\}.$$

$$\mathcal{S}_2 := \{U \in \mathcal{U} \ : \ U.c_1x > x_{med}\}.$$

$$\mathcal{S}_{12} := \{U \in \mathcal{U} \ : \ U.c_1x \leqslant x_{med} \leqslant U.c_2x\} \text{ and}$$

(*i*) the members of $\mathcal{S}_{12}$ induce a vertical U-graph, (*ii*) any $U_1 \in \mathcal{S}_1$ and $U_2 \in \mathcal{S}_2$ are independent. We compute and combine approximate solutions of $G[\mathcal{S}_1]$ and $G[\mathcal{S}_2]$ to get one candidate solution and compute a 2-approximate solution on $G[\mathcal{S}_{12}]$ to get a $2(\log n)$-approximate solution over $G[\mathcal{U}]$. This hinges on the fact (which can be verified easily) that Theorem 8 is also applicable to U-graphs. This is stated explicitly below.

Theorem 12. *Let $\alpha(n)$ be an arbitrary non-decreasing function of n. Suppose IndSet2($\mathcal{U}$) is an an efficient, $\alpha(n)$-approximate MIS algorithm over vertical U-graphs. Then, IndSet1($\mathcal{U}$) is an efficient, $\alpha(n)(\log n)$-approximate MIS algorithm over U-graphs. For both approximation algorithms, n stands for the size of the input.*

In the following subsection, we present an efficient and 2-approximate algorithm for vertical U-graphs. As a consequence, we obtain the following conclusion.

Theorem 13. *IndSet1($\mathcal{U}$) is an efficient, $2(\log n)$-approximate algorithm for MIS on U-graphs.*

Proof. Follows from Theorem 12 (by setting $\alpha(n) = 2$ for every n), since (as is shown in the following subsection) a 2-approximation of MIS on vertical U-graphs can be obtained in polynomial time. □

6.3.1 2-approximation of MIS on vertical U-graphs

We begin with some definitions.

Definition 17. *A set $\mathcal{U}$ of $\sqcup$-shaped objects is said to form a LU-*graph *(RU-*graph*) if the left (right) vertical arm is as long as the right (left) vertical arm, for each $U \in \mathcal{U}$.*

Definition 18. *A set $\mathcal{U}$ of $\sqcup$-shaped objects is said to form a* vertical LU-graph *if the induced graph is both a vertical U-graph and a LU-graph.* vertical RU-graphs *are similarly defined.*

Let $\mathcal{U}$ induce a vertical U-graph. We decompose it into $\mathcal{U} = \mathcal{U}_l \cup \mathcal{U}_r$ where $\mathcal{U}_l$ ($\mathcal{U}_r$) is the set of those $U \in \mathcal{U}$ whose left (right) vertical arm is as long as its right (left) vertical arm. Those Us for which the two vertical arms are of equal length are placed in both. We let $G = G[\mathcal{U}]$, $G_l = G[\mathcal{U}_l]$ and $G_r = G[\mathcal{U}_r]$. We also let I^*, I^*_l and I^*_r denote respectively an arbitrary but fixed MIS in each of G, G_l and G_r.

For each of the classes of vertical LU-graphs and vertical RU-graphs, we present an efficient and exact algorithm for finding a MIS in a given input from that class. Applying this to each of $G[\mathcal{U}_l]$ and $G[\mathcal{U}_r]$, we deduce that one can efficiently find a MIS for each of these graphs. We have either $|I^* \cap \mathcal{U}_l| \geqslant |I^*|/2$ or $|I^* \cap \mathcal{U}_r| \geqslant |I^*|/2$. We also have $|I^*_l| \geqslant |I^* \cap \mathcal{U}_l|$ and also $|I^*_r| \geqslant |I^* \cap \mathcal{U}_r|$. As a result, we have $\max\{|I^*_l|, |I^*_r|\} \geqslant |I^*|/2$. Thus, computing a MIS in each of G_l and G_r and choosing the best of these two solutions, gets us a 2-approximation to a MIS in G.

Since LU-graphs and RU-graphs are the same (since one can go from one representation to the other one by a reflection about the y-axis), it suffices to present an exact MIS algorithm for the class of vertical LU-graphs. This exact algorithm is based on Dynamic Programming (as in the case of vertical L-graphs) and is similarly based on a recursive computation of MIS. We have analogues of Definitions 5 through 11 and Lemma 21 for the case of U's inducing a vertical LU-graph. There are subtle differences in some of the analogous definitions in order to take into account the presence of a vertical arm at the right corner point.

Definition 19. *Let U, U' be two arbitrary $\sqcup$'s. We say that $U <_x U'$ if $U.c_1x < U'.c_1x$. We say that $U <_y U'$ if $U.c_1y > U'.c_1y$.*

Definition 20. *Let U, U' be two arbitrary $\sqcup$'s. We say that U' is* entirely right and below *of U if (i) $U <_x U'$, (ii) $U <_y U'$ and (iii) $U'.ly < U.c_2y$. We say that U' is* entirely right and above *of U if (i) $U <_x U'$, (ii) $U' <_y U$, (iii) either $U'.c_2x < U.c_2x$ or $U.ry < U'.c_2y$.*

Definition 21. *Let S be an arbitrary set of Us such that each member intersects a common vertical line $x = a$. A* (cap, cushion) *of S is any pair (U_1, U_2) of $\sqcup$'s each intersecting $x = a$ such that (i) each $U' \in S$ is entirely right and below of U_1, (ii) each $U' \in S$ is entirely right and above of U_2, (iii) U_2 is entirely right and below of U_1.*

Definition 22. *Let S be an arbitrary set of Us such that each member intersects a common vertical line $x = a$. Let (U_1, U_2) be a pair of U's also intersecting $x = a$ such that U_2 is entirely right and below of U_1. We define the subset of S capped and cushioned by*

(U_1, U_2) *to be the set of those* $U \in S$ *such that* (i) U *is entirely right and below of* U_1 *and* (ii) U *is entirely right and above of* U_2. *We denote this set by* S_{U_1,U_2}.

Definition 23. *Given a* $\mathcal{S}$ *and a cap* U, *and a* $U'' \in \mathcal{S} \cup \{U\}$, *we use* $\mathcal{S}^{U''}$ *to denote the subset of those* $U' \in \mathcal{S}$ *which are smaller or equal to* U'' *with respect to* $<_y$ *ordering, that is, the set* $\{U' \in \mathcal{S} : U' <_y U'' \vee U' = U''\}$. *In particular, we have* $\mathcal{S} = \mathcal{S}^{U_s}$ *always where* U_s *is the last element of* $\mathcal{S}$ *with respect to* $<_y$ *ordering. Also,* $\mathcal{S}^U = \varnothing$ *always.*

Definition 24. *For a set* $\mathcal{S}$ *capped and cushioned by* (U, U') *with* U_s *being the last element (with respect to* $<_y$ *ordering), let* $LA(\mathcal{S}, U_s)$ *denote the set of those* U'' *such that either* (i) $U'' = U$ *or* (ii) $U'' \in \mathcal{S} \setminus \{U_s\}$ *and* U_s *is entirely right and below* U''.

Definition 25. *For a set* $\mathcal{S}$ *inducing a vertical* LU*-graph* G, *capped and cushioned by* (U, U'), *we use* $Opt(\mathcal{S}, U, U')$ *to denote any MIS in* G.

Our algorithm is recursive and is based on the following recursion satisfied by $Opt(\mathcal{S}, U, U')$. The proof of this lemma is similar to that of Lemma 21 and is skipped.

Lemma 23. *Let* $\mathcal{S}, U, U'$ *be as in the previous definition with* U_s *being the last memebr of* $\mathcal{S}$ *with respect to* $<_y$ *ordering. Then,* $Opt(\mathcal{S}, U, U')$ *equals (in size)*

$$\max\left\{Opt(\mathcal{S} \setminus \{U_s\}, U, U'), \max_{U'' \in LA(\mathcal{S},U_s)} \left\{\{U_s\} \cup Opt(\mathcal{S}^{U''}, U, U') \cup Opt(\mathcal{S}_{U'',U_s}, U'', U_s)\right\}\right\},$$

provided $|\mathcal{S}| \geqslant 2$. *Otherwise,* $Opt(\mathcal{S}, U, U') = \mathcal{S}$.

Suppose $\mathcal{S}$ is a set of n members inducing a vertical LU-graph with $x = a$ being the common vertical line. Let $(U_1, U_2, \ldots, U_n)$ be the linear ordering of $\mathcal{S}$ defined by $U_i <_y U_j$ for every $i < j$. Choose a cap and cushion (U_0, U_{n+1}) for $\mathcal{S}$. It is easy to see that one can always compute such a pair in linear time. The correctness and the claim of polynomial time bound are based on the series of Claims $2 - 6$ presented in Subsection 5.1.1 . Let T denote the unique recursion tree capturing the recursion based computation of $Opt(\mathcal{S}, U_0, U_{n+1})$. Arguing as before (for the case of L-graphs), one can obtain the following theorem.

Theorem 14. *There exists an $O(n^4)$ time exact algorithm for finding a MIS in vertical LU-graphs.*

6.4 Approximation algorithms for Z-graphs

Z-graphs are graphs induced by a set of Z-shapes. Z-shapes are similar to U-shapes except that the right vertical arm of each Z is pointed down. This brings about a some new complications which need to be taken care of. As in the case of U-graphs, we reduce the problem of approximating a MIS over Z-graphs to the problem of approximating a MIS over vertical Z-graphs.

Definition 26. *A set $\mathcal{Z}$ of Z-shaped objects is said to form a* vertical Z*-graph if there exists a vertical line $x = a$ intersecting every $Z \in \mathcal{Z}$.*

Below, we present an algorithm which solves MIS over vertical Z-graphs within a multiplicative factor of $2(\log n)$. This, in turn (based on arguments similar to those employed for L- and U-graphs), leads to an algorithm for solving MIS over Z-graphs within a multiplicative factor of $2(\log n)^2$. As for computing the median and partitioning, the $Z.c_1x$ and $Z.c_2x$ values take respectively the roles of $L.cx$ and $L.hx$ values. As before, we compute a median x_{med} and partition $\mathcal{Z}$ into $\mathcal{Z} = \mathcal{Z}_1 \cup \mathcal{Z}_2 \cup \mathcal{Z}_{12}$ where

$$\mathcal{Z}_1 := \{Z \in \mathcal{Z} \ : \ Z.c_2x < x_{med}\}.$$

$$\mathcal{Z}_2 := \{Z \in \mathcal{Z} \ : \ Z.c_1x > x_{med}\}.$$

$$\mathcal{Z}_{12} := \{Z \in \mathcal{Z} \ : \ Z.c_1x \leqslant x_{med} \leqslant Z.c_2x\} \text{ and}$$

(i) the members of $\mathcal{Z}_{12}$ induce a vertical Z-graph, (ii) any $Z_1 \in \mathcal{Z}_1$ and $Z_2 \in \mathcal{Z}_2$ are independent. We compute and combine approximate solutions of $G[\mathcal{Z}_1]$ and $G[\mathcal{Z}_2]$ to get one candidate solution and compute a $2(\log n)$-approximate solution on $G[\mathcal{Z}_{12}]$ to

get a $2(\log n)^2$-approximate solution over $G[\mathcal{Z}]$. This hinges on the fact (which can be verified easily) that Theorem 8 is also applicable to Z-graphs. It now remains to present a $2(\log n)$-approximate algorithm for vertical Z-graphs.

6.4.1 $2(\log n)$-approximation of MIS on vertical Z-graphs

Let $G = (\mathcal{Z}, E)$ be a vertical Z-graph intersecting a common vertical line $x = a$. We sort the Z's based on their $Z.c_1y$ values. Let y_{med} be a median of these values. Partition $\mathcal{Z}$ into $\mathcal{Z} = \mathcal{W}_1 \cup \mathcal{W}_2 \cup \mathcal{W}_{12}$ where $\mathcal{W}_1 = \{Z \in \mathcal{Z} : Z.dy > y_{med}\}$, $\mathcal{W}_2 = \{Z \in \mathcal{Z} : Z.ty < y_{med}\}$ and $\mathcal{W}_{12} = \{Z \in \mathcal{Z} : Z.dy \leqslant y_{med} \leqslant Z.ty\}$. $\mathcal{W}_{12}$ is an example of vertical-horizontal (VH) Z-graphs. We have $|\mathcal{W}_1|, |\mathcal{W}_2| \leqslant n/2$ where $n = |\mathcal{Z}|$.

Definition 27. *A class $\mathcal{Z}$ of Z-shapes is said to induce a VH Z-graph if there exists a vertical line $x = a$ and a horizontal line $y = b$ such that each $Z \in \mathcal{Z}$ intersects both $x = a$ and $y = b$.*

As for L-graphs, U-graphs and even general Z-graphs, one can establish that existence of a $\alpha(n)$-approximate algorithm for VH Z-graphs leads to the existence of a $\alpha(n)(\log n)$-approximate algorithm for vertical Z-graphs. Thus, we prove that a $2(\log n)$-approximation of MIS can be efficiently obtained for vertical Z-graphs by proving that a 2-approximate solution of MIS for VH Z-graphs can be obtained efficiently.

To achieve a 2-approximation of MIS over VH Z-graphs, we decompose the input $\mathcal{Z}$ of any such graph (with each $Z \in \mathcal{Z}$ intersecting both $x = a$ and $y = b$) into $\mathcal{Z} = \mathcal{Z}^t \cup \mathcal{Z}^d$ where $\mathcal{Z}^t$ is the set of those $Z \in \mathcal{Z}$ whose top vertical arm intersects $y = b$ and $\mathcal{Z}^d$ is the set of those $Z \in \mathcal{Z}$ whose bottom vertical arm intersects $y = b$. When the horizontal arm of a Z lies on $y = b$, such a Z is a member of $\mathcal{Z}^t$ and $\mathcal{Z}^d$. We call the class of graphs induced by $\mathcal{Z}^t$s as vertical-t-horizontal (VtH) Z-graphs. Similarly, we call the class of graphs induced by $\mathcal{Z}^d$s as vertical-d-horizontal (VdH) Z-graphs. Formally,

Definition 28. *A class $\mathcal{Z}$ of Z-shapes is said to induce a VtH (or VdH) Z-graph if there exists a vertical line $x = a$ and a horizontal line $y = b$ such that for each $Z \in \mathcal{Z}$, its top (bottom) vertical arm intersects $y = b$ and its horizontal arm intersects $x = a$.*

In the next subsection, we show that for each of these two classes of graphs, an MIS can be computed exactly and efficiently. We present the arguments only for the class of VtH Z-graphs. The arguments are similar for the case of VdH Z-graphs. We solve MIS exactly for each of the subgraphs induced by $\mathcal{Z}^t$ and $\mathcal{Z}^d$ and choose the best of the two solutions. This gives us a 2-approximation of MIS in the subgraph induced by $\mathcal{W}_{12}$.

6.4.2 Exact computation of MIS over VtH Z-graphs

This exact algorithm is based on Dynamic Programming (as in the cases of vertical L, LU-graphs) and is similarly based on a recursive computation of MIS. We have analogues of Definitions 5 through 11 and Lemma 21 for the case of Z's inducing a VtH Z-graph.

Definition 29. *Let Z, Z' be two arbitrary Z's. We say that $Z <_x Z'$ if $Z.c_1x < Z'.c_1x$. We say that $Z <_y Z'$ if* either *(i) $Z.c_1y > Z'.c_1y$* or *(ii) $Z.c_1y = Z'.c_1y$ and $Z.c_1x > Z'.c_1x$.*

Definition 30. *Let Z, Z' be two arbitrary Z's. We say that Z is* entirely left and below *of Z' if (i) $Z.c_1 < Z'.c_1$ and (ii) either $Z.c_2 < Z'.c_2$ or $Z'.dy > Z.c_2y$. We say that Z is* entirely right and above *of Z' if Z' is entirely left and below of Z.*

Definition 31. *Let S be an arbitrary set of Zs such that each member intersects a common vertical line $x = a$ and also a common horizontal line $y = b$ with its top vertical arm. A* (cap, cushion) *of S is any pair (Z_1, Z_2) of Z's each intersecting $x = a$ and $y = b$ (with its top vertical arm) such that (i) each $Z' \in S$ is entirely left and below of Z_1, (ii) Z_2 is entirely left and below of Z' for each $Z' \in S$. (iii) Z_2 is entirely left and below of Z_1.*

Definition 32. *Let S be an arbitrary set of Zs such that each member intersects a common vertical line $x = a$ and also a common horizontal line $y = b$ with its top vertical arm. Let*

(Z_1, Z_2) be a pair of Z's each intersecting $x = a$ and $y = b$ (with its top vertical arm) such that Z_2 is entirely left and below of Z_1. We define the subset of S capped and cushioned by (Z_1, Z_2) to be the set of those $Z \in S$ such that (i) Z is entirely left and below of Z_1 and (ii) Z is entirely right and above of Z_2. We denote this set by S_{Z_1,Z_2}.

Definition 33. *Given a $\mathcal{S}$ and a cap Z, and a $Z'' \in \mathcal{S} \cup \{Z\}$, we use $\mathcal{S}^{Z''}$ to denote the subset of those $Z' \in \mathcal{S}$ which are smaller or equal to Z'' with respect to $<_y$ ordering, that is, the set $\{Z' \in \mathcal{S} : Z' <_y Z'' \vee Z' = Z''\}$. In particular, we have $\mathcal{S} = \mathcal{S}^{Z_s}$ always where Z_s is the last element of $\mathcal{S}$ with respect to $<_y$ ordering. Also, $\mathcal{S}^Z = \emptyset$ always.*

Definition 34. *For a set $\mathcal{S}$ capped and cushioned by (Z, Z') with Z_s being the last element (with respect to $<_y$ ordering), let $LA(\mathcal{S}, Z_s)$ denote the set of those $Z'' \in \mathcal{S} \cup \{Z\}$ such that either (i) $Z'' = Z$ or (ii) $Z'' \in \mathcal{S} \setminus \{Z_s\}$ and Z_s is entirely right and below Z''.*

Definition 35. *For a set $\mathcal{S}$ inducing a VtH Z-graph G, capped and cushioned by (Z, Z'), we use $Opt(\mathcal{S}, Z, Z')$ to denote any MIS in G.*

Our algorithm is recursive and is based on the following recursion satisfied by $Opt(\mathcal{S}, Z, Z')$. The proof of this lemma employs arguments similar to those of Lemma 21 and is skipped.

Lemma 24. *Let $\mathcal{S}, Z, Z'$ be as in the previous definition with Z_s being the last memebr of $\mathcal{S}$ with respect to $<_y$ ordering. Then, $Opt(\mathcal{S}, Z, Z')$ equals (in size)*

$$\max\left\{Opt(\mathcal{S} \setminus \{Z_s\}, Z, Z'), \{Z_s\} \cup Opt(\mathcal{S}_{Z,Z_s}, Z, Z_s)\right\},$$

provided $|\mathcal{S}| \geqslant 2$. Otherwise, $Opt(\mathcal{S}, Z, Z') = \mathcal{S}$.

Suppose $\mathcal{S}$ is a set of n members inducing a VtH Z-graph with $x = a$ being the common vertical line and $y = b$ being the common horizontal line being intersected by every member of Z. Let $(Z_1, Z_2, \ldots, Z_n)$ be the linear ordering of $\mathcal{S}$ defined by $Z_i <_y Z_j$ for every $i < j$. Choose a cap and cushion (Z_0, Z_{n+1}) for $\mathcal{S}$. It is easy to see that one can always compute such a pair in linear time. The correctness and the claim of polynomial

time bound are based on the series of following Claims (whose proofs are presented in the appendix) and are analogous to Claims 1-5. Let T denote the unique recursion tree capturing the recursion based computation of $Opt(\mathcal{S}, Z_0, Z_{n+1})$.

Claim 13. *The problem size ($|\mathcal{S}|$) keeps decreasing along every path in T until it reaches the base case* $|\mathcal{S}| \leqslant 1$.

Proof. Each of the sets $\mathcal{S} \setminus \{Z_s\}, \mathcal{S}_{Z,Z_s}$ has a size which is less than that of $\mathcal{S}$. □

Claim 14. *Each of the sets $\mathcal{S}$ defining a subproblem is a subset of the original input* $\{Z_1, \ldots, Z_n\}$.

Proof. The proof is based on the depth of recursion from the root of T. The claim is obviously true for the root. Each of the sets $\mathcal{S} \setminus \{Z_s\}, \mathcal{S}_{Z,Z_s}$ is a subset of $\mathcal{S}$ which is the input for the current subproblem. □

Claim 15. *Every pair (Z, Z') of (cap,cushion) that arises in any subproblem generated by the above recursion is of the form* (Z_0, Z_j) *where* $0 < j \leqslant n + 1$.

Proof. The proof is based on the depth of recursion from the root of T. The claim is obviously true for the root. Suppose it is true for some problem corresponding to a node in T. If (Z, Z') is the pair corresponding to this problem, then for each of its child subproblems, the pair is either (Z, Z') or (Z, Z_s). This proves the claim. □

Claim 16. *Each of the sets $\mathcal{S}'$ defining a subproblem is a set of the form $\mathcal{S}^{Z_k}_{Z_0,Z_j}$ for some* $0 < k < j \leqslant n + 1$ *and* $\mathcal{S} = \{Z_1, \ldots, Z_n\}$.

As a consequence, we obtain the following corollary.

Claim 17. *There are at most n^2 distinct subproblems that are actually solved in the recursion formulation.*

Arguing as before (for the case of L-graphs), one can obtain the following theorem.

Theorem 15. *There exists an $O(n^3)$ time exact algorithm for finding a MIS in VtH Z-graphs.*

6.5 Appendix 2: Proof of Assumption (2) :

The proof outline of Assumption (2) is similar to that of Assumption (1) given before but there are some complications which arise and need to be taken care of, on account of the presence of a vertical arm at the right tip of the horizontal arm. In particular, the U shapes are not symmetrical with respect to the x and y-axes. Hence, one needs to work-out a separate proof for ensuring that $U.c_1y \neq U'.c_1y$ (for every $U \neq U'$).

As before, we say that two sets $\mathcal{U}, \mathcal{U}''$ of U's are *equivalent* if $G[\mathcal{U}]$ and $G[\mathcal{U}'']$ are isomorphic. We prove Assumption (2) in two steps as follows : First, we prove that there exists an efficiently computable and equivalent (to $\mathcal{U}$) $\mathcal{U}' = \{U'_1, \ldots, U'_n\}$ such that (i) $U'_i.c_1y = U_i.c_1y$ for each i and (ii) $U'_i.c_1x \neq U'_j.c_1x$ for every $i \neq j$. Then, by separate arguments, we also prove that there exists an efficiently computable and equivalent (to $\mathcal{U}'$) $\mathcal{U}'' = \{U''_1, \ldots, U''_n\}$ such that (i) $U''_i.c_1x = U'_i.c_1x$ for each i and (ii) $U''_i.c_1y \neq U''_j.c_1y$ for every $i \neq j$. $\mathcal{U}''$ is the required set of U's. Recall that we assume that the left vertical arm of each $U \in \mathcal{U}$ is as long as its right vertical arm.

6.5.1 Existence and computation of $\mathcal{U}'$

The existence and computation of $\mathcal{U}'$ follows from the following two claims.

Claim 18. *For every finite set $\mathcal{U} = \{U_i\}_i$ of U's, there exists an efficiently computable and equivalent $\mathcal{U}^a = \{U^a_i\}_i$ such that (A) $U^a_i.c_2x \neq U^a_j.c_1x$ for every $i \neq j$, (B) $U^a_i.c_1y = U_i.c_1y$ for every i and (C) vertical and horizontal arms of U^a_i have the same respective lengths as those of U_i, for every i.*

Proof. The proof is essentially that of Claim 11 with some subtle and important changes to take care of the right vertical arm, with $U.c_1x$ and $U.c_2x$ taking respectively the roles of $L.cx$ and $L.hx$ for every U. Let $(x_k)_k$, $(\alpha_k)_k$, $(\beta_i)_i$, α, β and γ be defined as in the proof of Claim 11 with $\mathcal{L}$ being the set of L's one obtains by replacing each U by its corresponding L.

For every i, define U_i^a as the result of shifting U_i in the negative x-direction by $k\gamma$ distance, where $x_k = U_i.c_1x$. That is, U_i^a is characterized by $(x_k{-}k\gamma, U_i.c_1y, U_i.c_2x{-}k\gamma, U_i.ly, U_i.ry)$. That $\mathcal{U}^a$ satisfies condition (A) has been established in the proof of Claim 11.

We establish that $\mathcal{U}^a$ and $\mathcal{U}$ are equivalent by establishing that for every $i \neq j$, U_i^a and U_j^a are independent if and only if U_i and U_j are independent. Fix an arbitrary $i \neq j$. Without loss of generality, assume that $U_i.c_1x \leqslant U_j.c_1x$.

Suppose, $U_i.c_1x = U_j.c_1x = x_k$. Then, $U_i^a.c_1x = U_j^a.c_1x = x_k - k\gamma$. Clearly, U_i^a and U_j^a intersect if and only if U_i and U_j intersect. Hence, from now onwards, assume that $U_i.c_1x < U_j.c_1x$. Let $U_j.c_1x = x_k$ where $k \geqslant 2$.

If $U_i.c_2x < U_j.c_1x$, then U_i and U_j are independent. Also, $U_j^a.c_1x = U_j.c_1x - k\gamma \geqslant U_j.c_1x - \frac{\beta}{2} > U_i.c_2x \geqslant U_i^a.c_2x$ and hence U_i^a and U_j^a are independent.

Suppose we have $U_j.c_1x \leqslant U_i.c_2x$. We have $U_j^a.c_1x = x_k - k\gamma \geqslant x_k - \frac{\alpha}{2} > x_{k-1} \geqslant U_i.c_1x \geqslant U_i^a.c_1x$. If $U_j.c_1x < U_i.c_2x$, then $U_i^a.c_2x \geqslant U_i.c_2x - k\gamma \geqslant U_i.c_2x - \frac{\beta}{2} > U_j.c_1x > U_j^a.c_1x$. If $U_j.c_1x = U_i.c_2x$, then $U_i^a.c_2x \geqslant x_k - (k-1)\gamma > x_k - k\gamma = U_j^a.c_1x$. In any case, we have $U_i^a.c_1x \leqslant U_j^a.c_1x \leqslant U_i^a.c_2x$. Now consider the following cases which exhaust all possibilitites.

(i) $U_j.ly < U_i.c_1y$ which happens if and only if $U_j^a.ly < U_i^a.c_1y$. In that case, we have U_i and U_j are independent as also U_i^a and U_j^a.

(ii) $U_j.c_1y \leqslant U_i.c_1y \leqslant U_j.ly$ which happens if and only if $U_j^a.c_1y \leqslant U_i^a.c_1y \leqslant U_j^a.ly$. In that case, we have U_i and U_j intersect as also U_i^a and U_j^a.

(*iii*) $U_i.c_1y < U_j.c_1y$ which happens if and only if $U_i^a.c_1y < U_j^a.c_1y$. We have two further subcases.

(*iii*)(*a*) $U_j.c_2x < U_i.c_2x$. In this case, U_i and U_j are independent. Also, $U_j^a.c_2x = U_j.c_2x - k\gamma < U_j.c_2x - (k-1)\gamma < U_i.c_2x - (k-1)\gamma \leqslant U_i^a.c_2x$. Hence, U_i^a and U_j^a are also independent.

(*iii*)(*b*) $U_i.c_2x \leqslant U_j.c_2x$. Hence, U_i and U_j intersect if and only if $U_i.ry \geqslant U_j.c_1y$. Also, if $U_i.c_2x > U_j.c_1x$, then $U_i^a.c_2x > U_j.c_1x > U_j^a.c_1x$. Otherwise, if $U_i.c_2x = U_j.c_1x$, then $U_i^a.c_2x \geqslant U_i.c_2x - (k-1)\gamma > U_j.c_1x - k\gamma = U_j^a.c_1x$. Hence, U_i^a and U_j^a intersect if and only if $U_i^a.ry \geqslant U_j^a.c_1y$ which happens if and only if $U_i.ry \geqslant U_j.c_1y$. □

Claim 19. *For every $\mathcal{U}^a$ mentioned before, there exists an efficiently computable and equivalent $\mathcal{U}' = \{U_i'\}_i$ such that (D) $U_i'.c_1x \neq U_j'.c_1x$ for every $i \neq j$, (E) $U_i'.c_1y = U_i^a.c_1y$ for every i and (F) vertical and horizontal arms of U_i^a have the same respective lengths as those of U_i' for every i.*

Proof. Let $(x_k)_k$, $(\alpha_k)_k$, $(\beta_i)_i$, α and β be defined as in the proof of Claim 18 with $\mathcal{U} = \mathcal{U}^a$. For every i, define δ_i to be $U_i^a.c_2x - U_i^a.c_1x$ and define δ to be $\min_i \delta_i$. In addition, let $y_1 < y_2 < \ldots < y_r < \infty = y_{r+1}$ be the sorted list of $U_i^a.c_2x$ values. Define $\epsilon = \min_{i \leqslant r}\{y_{i+1} - y_i\}$. Redefine γ to be $\min\{\frac{\alpha}{2n}, \frac{\beta}{2n}, \frac{\delta}{2n}, \frac{\epsilon}{2n}\}$ where $n = |\mathcal{U}|$. We have $\gamma > 0$.

Define $\mathcal{U}'$ in terms of $\mathcal{U}^a$ as follows. Fix an arbitrary k and order all those U's in $\mathcal{U}^a$ having $U.c_1x = x_k$ as follows. If U_1^a, U_2^a are two such members, then $U_1^a < U_2^a$ (or vice versa) if $U_1^a.c_1y > U_2^a.c_1y$. If $U_1^a.c_1y = U_2^a.c_1y$, then $U_1^a < U_2^a$ (or vice versa) if $U_1^a.c_2x < U_2^a.c_2x$. If $U_1^a.c_1y = U_2^a.c_1y$ and $U_1^a.c_2x = U_2^a.c_2x$, then $U_1^a < U_2^a$ (or vice versa) if $U_1^a.ly < U_2^a.ly$. If $U_1^a.c_1y = U_2^a.c_1y$, $U_1^a.c_2x = U_2^a.c_2x$ and $U_1^a.ly = U_2^a.ly$, then $U_1^a < U_2^a$ (or vice versa) if $U_1^a.ry < U_2^a.ry$. Surely, any pair of distinct U's differ in at least one of the five values. Let O_k be the resulting total ordering obtained. Now, concatenate all such orderings obtained (one for each k) as $O_1, O_2, \ldots, O_m$ where m is the number of distinct values of $U^a.c_1x$ ($U^a \in \mathcal{U}^a$). Let $(U_1^a, \ldots, U_n^a)$ be the resulting total ordering of $\mathcal{U}^a$.

For every $i \leqslant n$, define U'_i as the U characterized by $(U^a_i.c_1x + i\gamma, U^a_i.c_1y, U^a_i.c_2x + i\gamma, U^a_i.c_2y, U^a_i.ly, U^a_i.ry)$. Note that $U'_i.c_1x - U^a_i.c_1x \leqslant n\gamma \leqslant \min\{\frac{\alpha}{2}, \frac{\beta}{2}, \frac{\delta}{2}, \frac{\epsilon}{2}\}$ for every i. Similarly, $U'_i.c_2x - U^a_i.c_2x \leqslant n\gamma \leqslant \min\{\frac{\alpha}{2}, \frac{\beta}{2}, \frac{\delta}{2}, \frac{\epsilon}{2}\}$ for every i.

That Condition (D) is satisfied by $\mathcal{U}'$ can be seen as follows. Fix an arbitrary $i < j$ satisfying $U^a_i.c_1x = x_r \leqslant x_t = U^a_j.c_1x$. If $x_r < x_t$, then $U'_i.c_1x \leqslant x_r + \frac{\alpha}{2} < x_t \leqslant U'_j.c_1x$. If $x_r = x_t$, it follows from the definition of the ordering associated with x_r that $U'_i.c_1x < U'_j.c_1x$.

We only need to show that for every $i < j$, U'_i and U'_j are independent if and only if U^a_i and U^a_j are independent. Fix an arbitrary $i < j$. It follows that $U^a_i.c_1x \leqslant U^a_j.c_1x$.

Case 1 : $U^a_i.c_1x = U^a_j.c_1x = x_k$. Then, $U'_i.c_1x < U'_j.c_1x \leqslant x_k + \frac{\delta}{2} < U^a_i.c_2x \leqslant U'_i.c_2x$. There are two sub-cases.

(i) Suppose $U^a_i.c_1y > U^a_j.c_1y$. Then, U^a_i and U^a_j intersect if and only if $U^a_i.c_1y \leqslant U^a_j.ly$. Also, U'_i and U'_j intersect if and only if $U'_i.c_1y = U^a_i.c_1y \leqslant U'_j.ly = U^a_j.ly$, since $U'_i.c_1x < U'_j.c_1x < U'_i.c_2x$. Thus, U^a_i and U^a_j intersect if and only if U'_i and U'_j intersect.

(ii) Suppose $U^a_i.c_1x = U^a_j.c_1x = x_k$ and $U^a_i.c_1y = U^a_j.c_1y$. Then, U^a_i and U^a_j intersect. Also, $U'_i.c_1x \leqslant U'_j.c_1x < U'_i.c_2x$ as shown before. Hence, U'_i and U'_j also intersect.

Case 2 : Suppose $U^a_i.c_1x = x_r < x_t = U^a_j.c_1x$. If $U^a_i.c_2x < U^a_j.c_1x$, then U^a_i and U^a_j are independent. Also, $U'_i.c_2x \leqslant U^a_i.c_2x + \frac{\beta}{2} < U^a_j.c_1x \leqslant U'_j.c_1x$. Hence, U'_i and U'_j are independent.

The only remaining case (follows from Assumption (A) satisfied by $\mathcal{U}^a$) is that (i) $U^a_i.c_1x = x_r < x_t = U^a_j.c_1x < U^a_i.c_2x$. In that case, (ii) $U'_i.c_1x \leqslant x_r + \frac{\alpha}{2} < x_t \leqslant U'_j.c_1x \leqslant x_t + \frac{\beta}{2} < U^a_i.c_2x \leqslant U'_i.c_2x$.

In view of (i), U^a_i and U^a_j intersect if and only if *either* (a) $U^a_j.c_1y \leqslant U^a_i.c_1y \leqslant U^a_j.ly$ *or* (b) $U^a_i.c_1y < U^a_j.c_1y$ and $U^a_i.c_2x \leqslant U^a_j.c_2x$ and $U^a_i.ry \geqslant U^a_j.c_1y$.

In view of (ii), U'_i and U'_j intersect if and only if *either* (a) $U'_j.c_1y \leqslant U'_i.c_1y \leqslant U'_j.ly$ *or* (b)

$U'_i.c_1y < U'_j.c_1y$ and $U'_i.c_2x \leqslant U'_j.c_2x$ and $U'_i.ry \geqslant U'_j.c_1y$.

In addition, if $U^a_j.c_2x < U^a_i.c_2x$, then $U'_j.c_2x = U^a_j.c_2x + \frac{\epsilon}{2} < U^a_i.c_2x \leqslant U'_i.c_2x$. If $U^a_i.c_2x \leqslant U^a_j.c_2x$, then $U'_i.c_2x = U^a_i.c_2x + i\gamma < U^a_j.c_2x + j\gamma = U'_j.c_2x$. Thus, we notice that $U^a_i.c_2x \leqslant U^a_j.c_2x$ if and only if $U'_i.c_2x \leqslant U'_j.c_2x$.

Hence, (a) (or (b)) happens between U^a_i and U^a_j if and only if (a) (or (b)) happens between U'_i and U'_j. Hence U^a_i and U^a_j intersect if and only if U'_i and U'_j intersect. This establishes that $\mathcal{U}^a$ is equivalent to $\mathcal{U}'$. □

Chapter 7

Conclusions

7.1 Summary

In this thesis, we studied two algorithmic problems. They are *Stochastic Matching* over general graphs and *Maximum Independent Set (MIS)* over B_1, B_2-VPG graphs. We first presented our results for the stochastic matching problem and then we presented our results for MIS problem over B_1, B_2-VPG graphs.

In Chapter 1, we analyzed the performance of the greedy algorithm for weighted stochastic matching. We obtain an approximation guarantee of $2/p_{min}^2$. Here p_{min} is the minimum probability associated with any edge in the input. Since p_{min} can become arbitrarily small asymptotically, this implies that performance guarantee of the greedy algorithm can become unbounded. We also presented an infinite and explicit family of weighted graphs for which it is shown that the approximation ratio of the greedy algorithm is at least $2/p_{min}$.

In Chapter 3, we presented an algorithm (based on an LP formulation and an application of a randomized rounding procedure) for online stochastic matching. We analyzed this algorithm and established that its approximation ratio is at most 5.2.

The latter part of the thesis presents results obtained for the MIS problem over B_1, B_2-

VPG graphs. In Chapter 4, we present an $(\log n)^2$-approximate algorithm for the MIS problem over B_1-VPG graphs. For equilateral B_1-VPG graphs, we obtained a $36(\log 2d)$-approximate algorithm where d is the ratio of the maximum length to minimum lengths of any arm of any member of the input. We also proved that the MIS problem on unit L-graphs is NP-complete. Later, in Chapter 5, we present another algorithm (for MIS over B_1-VPG graphs) with an improved performance guarantee of $O(\log n)$. In Chapter 6, we present an approximate MIS algorithm for B_2-VPG graphs and prove that its approximation ratio is $O((\log n)^2)$.

7.2 Future Directions

In this section, we outline some potential future directions based on the work of the thesis.

For the greedy analysis of the weighted instance of stochastic matching, it would be interesting to reduce the gap between lower and upper bounds.

Another interesting problem is to design a combinatorial algorithm for weighted stochastic matching with a constant approximation factor.

A $(1+\epsilon)$-approximation algorithm with running time being at most $2^{poly(\log n, 1/\epsilon)}$ is said to be a *quasi polynomial time approximation scheme (QPTAS)*. Here n denotes the size of the input instance. A $(1+\epsilon)$-approximation algorithm with running time being $poly(n, 1/\epsilon)$ is said to be a *polynomial time approximation scheme (PTAS)*.

For B_1, B_2-VPG graphs, that MIS problem has a QPTAS follows from the works of Adamaszek et al and also from the works of Harpeled [AW14, HP14]. A related goal would be to design a PTAS for B_1, B_2-VPG graphs. The best known algorithm for MIS on string graphs has an approximation factor of n^ϵ for some $\epsilon > 0$ [FP11]. Another interesting direction is to explore the approximability of MIS on B_kVPG graphs with an approximation factor which is independent of k and is better than n^ϵ.

Bibliography

[ACG+12] Andrei Asinowski, Elad Cohen, Martin Charles Golumbic, Vincent Limouzy, Marina Lipshteyn, and Michal Stern. Vertex intersection graphs of paths on a grid. *J. Graph Algorithms Appl.*, 16(2):129–150, 2012.

[Ada11] Marek Adamczyk. Improved analysis of the greedy algorithm for stochastic matching. *Information Processing Letters*, 111(15):731–737, 2011.

[AFP98] Jonathan Aronson, Alan Frieze, and Boris G Pittel. Maximum matchings in sparse random graphs: Karp-sipser revisited. *Random Structures and Algorithms*, 12(2):111–177, 1998.

[AGM15] Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. Improved approximation algorithms for stochastic matching. In *Algorithms-ESA 2015*, pages 1–12. Springer, 2015.

[AM04] Pankaj K Agarwal and Nabil H Mustafa. Independent set of intersection graphs of convex objects in 2d. In *Algorithm Theory-SWAT 2004*, pages 127–137. Springer, 2004.

[ASW13] Marek Adamczyk, Maxim Sviridenko, and Justin Ward. Submodular stochastic probing on matroids. *arXiv preprint arXiv:1310.4415*, 2013.

[AVKS98] Pankaj K Agarwal, Marc Van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry*, 11(3):209–218, 1998.

[AW13] Anna Adamaszek and Andreas Wiese. Approximation schemes for maximum weight independent set of rectangles. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 400–409. IEEE, 2013.

[AW14] Anna Adamaszek and Andreas Wiese. A qptas for maximum weight independent set of polygons with polylogarithmically many vertices. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 645–656. SIAM, 2014.

[Bak94] Brenda S Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM (JACM)*, 41(1):153–180, 1994.

[BD15] Therese C. Biedl and Martin Derka. 1-string b_2-VPG representation of planar graphs. In *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, pages 141–155, 2015.

[BDMR01] Piotr Berman, Bhaskar DasGupta, S Muthukrishnan, and Suneeta Ramaswami. Efficient approximation algorithms for tiling and packing problems with rectangles. *Journal of Algorithms*, 41(2):443–470, 2001.

[BGL+10] Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When lp is the cure for your matching woes: improved bounds for stochastic matchings. In *Algorithms–ESA 2010*, pages 218–229. Springer, 2010.

[BK10] Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *Algorithms–ESA 2010*, pages 170–181. Springer, 2010.

[BV04] Stephen Poythress Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[CC09] Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 892–901. Society for Industrial and Applied Mathematics, 2009.

[CCS11] Steven Chaplick, Elad Cohen, and Juraj Stacho. Recognizing some subclasses of vertex intersection graphs of 0-bend paths in a grid. In *Graph-Theoretic Concepts in Computer Science - 37th International Workshop, WG 2011, Teplá Monastery, Czech Republic, June 21-24, 2011. Revised Papers*, pages 319–330, 2011.

[CGTW15] Elad Cohen, Martin Charles Golumbic, William T. Trotter, and Ruidong Wang. Posets and VPG graphs. *Order*, pages 1–11, 2015.

[CHP12] Timothy M Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012.

[CIK+09] Ning Chen, Nicole Immorlica, Anna R Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In *Automata, Languages and Programming*, pages 266–278. Springer, 2009.

[CJKV12] Steven Chaplick, Vít Jelínek, Jan Kratochvíl, and Tomás Vyskocil. Bend-bounded path intersection graphs: Sausages, noodles, and waffles on a grill. In *Graph-Theoretic Concepts in Computer Science - 38th International Workshop, WG 2012, Jerusalem, Israel, June 26-28, 2012, Revised Selcted Papers*, pages 274–285, 2012.

[CKU13] Steven Chaplick, Stephen G. Kobourov, and Torsten Ueckerdt. Equilateral L-contact graphs. In *Graph-Theoretic Concepts in Computer Science - 39th International Workshop, WG 2013, Lübeck, Germany, June 19-21, 2013, Revised Papers*, pages 139–151, 2013.

[CTT12] Kevin P Costello, Prasad Tetali, and Pushkar Tripathi. Stochastic matching with commitment. In *Automata, Languages, and Programming*, pages 822–833. Springer, 2012.

[CU12] Steven Chaplick and Torsten Ueckerdt. Planar graphs as vpg-graphs. In *Graph Drawing*, pages 174–186. Springer, 2012.

[CU13] Steven Chaplick and Torsten Ueckerdt. Planar graphs as VPG-graphs. *J. Graph Algorithms Appl.*, 17(4):475–494, 2013.

[DF91] Martin Dyer and Alan Frieze. Randomized greedy matching. *Random Structures & Algorithms*, 2(1):29–45, 1991.

[DFP93] Martin Dyer, Alan Frieze, and Boris Pittel. The average performance of the greedy matching algorithm. *The Annals of Applied Probability*, 3(2):526–552, 1993.

[Edm65] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965.

[FF56] Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[FKMU14] Stefan Felsner, Kolja B. Knauer, George B. Mertzios, and Torsten Ueckerdt. Intersection graphs of L-shapes and segments in the plane. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 299–310, 2014.

[FMMM09] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 117–126. IEEE, 2009.

[FP11] Jacob Fox and János Pach. Computing the independence number of intersection graphs. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pages 1161–1165, 2011.

[FRS93] Alan Frieze, AJ Radcliffe, and Stephen Suen. Analysis of a simple greedy matching algorithm on random cubic graphs. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 341–351. Society for Industrial and Applied Mathematics, 1993.

[GJ77] Michael R Garey and David S. Johnson. The rectilinear steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences. W. H. Freeman and Company, 1979.

[GKPS06] Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM (JACM)*, 53(3):324–360, 2006.

[GN13] Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In *Integer Programming and Combinatorial Optimization*, pages 205–216. Springer, 2013.

[Gol04] Martin C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Elsevier, 2004.

[GS62] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69(1):9–15, 1962.

[Hal95] Magnús M Halldórsson. Approximating discrete collections via local improvements. In *SODA*, volume 95, pages 160–169, 1995.

[Hås96] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 627–636. IEEE, 1996.

[HK73] John E Hopcroft and Richard M Karp. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[HMR+98] Harry B Hunt, Madhav V Marathe, Venkatesh Radhakrishnan, Shankar S Ravi, Daniel J Rosenkrantz, and Richard E Stearns. NC-approximation schemes for NP-and PSPACE-hard problems for geometric graphs. *Journal of Algorithms*, 26(2):238–274, 1998.

[HP14] Sariel Har-Peled. Quasi-polynomial time approximation scheme for sparse subsets of polygons. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 120. ACM, 2014.

[KH78] Bernhard Korte and Dirk Hausmann. An analysis of the greedy heuristic for independence systems. *Annals of Discrete Mathematics*, 2:65–74, 1978.

[KMP98] Sanjeev Khanna, S Muthukrishnan, and Mike Paterson. On approximating rectangle tiling and packing. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, volume 95, page 384. SIAM, 1998.

[KN90] Jan Kratochvíl and Jaroslav Nesetril. INDEPENDENT SET and CLIQUE problems in intersection-defined classes of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 031(1):85–93, 1990.

[MGS12] Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4):559–573, 2012.

[MR11] Marco Molinaro and R Ravi. The query-commit problem. *arXiv preprint arXiv:1110.0990*, 2011.

[Nie00] Frank Nielsen. Fast stabbing of boxes in high dimensions. *Theoretical Computer Science*, 246(1):53–72, 2000.

[Sch90] Walter Schnyder. Embedding planar graphs on the grid. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, 22-24 January 1990, San Francisco, California.*, pages 138–148, 1990.

[WS11] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.