Approximation Algorithms under the Worst-case Analysis and the Smoothed Analysis

by

Weitian Tong

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

How to evaluate the performance of an algorithm is a very important subject in computer science, for understanding its applicability, for understanding the problem which it is applied to, and for the development of new ideas that help to improve the existing algorithms. There are two main factors, *i.e.* the performance measure and the analysis model, that affect the evaluation of an algorithm. In this thesis, we analyse algorithms with approximation ratio as a measure under the worst-case analysis and the smoothed analysis. In particular, several interesting discrete optimization problems from bioinformatics, networking and graph theory are investigated, and currently the best approximation algorithms are designed for them under the classic analysis model — the worst-case analysis and the more "realistic" model — the smoothed analysis.

Acknowledgements

It would not have been possible to write this thesis without the support and encouragement of many people, to whom I wish to express my sincere appreciation.

Most of all, I thank my supervisor, Dr. Guohui Lin, for his invaluable guidance and support. It is difficult to summarize how many ways I am grateful to Dr. Lin, who is more like a close friend than a supervisor. Of course, I thank him mostly for his enthusiasm and for always leading my research into the right direction while leaving me a lot of freedom at the same time. I also thank him for his patience, understanding and advice as I balanced my research with other aspects of my life.

I would like to express my appreciation to my committee members, Dr. Randy Goebel, Dr. Russell Greiner, Dr. Guohui Lin, Dr. Mohammad R. Salavatipour, Dr. Csaba Szepesvári and Dr. Qianping Gu for their time, effort, and precious suggestions that helped my research to a great extend. Additionally, I would like to thank my collaborators, Dr. Randy Goebel, Dr. Guohui Lin, Dr. Tian Liu, Dr. Taibo Luo, Dr. Huili Zhang, Dr. Binhai Zhu for their help in my research.

I would also like to thank my references Dr. Randy Geobel, Dr. Guohui Lin, Dr. Tian Liu, Dr. Liang Zhao and Dr. Binhai Zhu for supporting me when I was applying for jobs. For personal funding and research grants, I thank the Department of Computing Science at the University of Alberta and Alberta Innovates Technology Futures (AITF) Graduate Studentship. And I also thank Dr. Randy Goebel and Dr. Guohui Lin for their generous funding support from the NSERC Discovery Grants.

Moreover, my appreciation goes to my friends and colleagues for their generous support and great companionship, especially the members in the Lin group. Last but not least, I want to express my gratitude and deepest appreciation to my family, especially my parents and my grandparents, for their continuous encouragement and wholehearted support. No matter where I am and how far away from them, they are always there for me. It is my great fortune in life to have them as my source of strength, happiness and love.

Contents

[1] This chapter is based on [53, 98, 99].
[2] This chapter is based on [103, 104].

[3]This chapter is based on [25].
[4]This chapter is based on [105].
[5]This chapter is based on [100, 101].
[6]This chapter is based on [102].

List of Figures

Chapter 1

Introduction

1.1 Roadmap

In this chapter, we will first introduce some basic definitions and notations that are extensively used in the thesis. Then we will briefly introduce the background about the approximation algorithm and analysis models.

In the following chapters, we will introduce several interesting problems and the approximation algorithms we designed. First we focus on the some interesting discrete optimization problems from bioinformatics, networking and graph theory, and design currently the best approximation algorithms for them under the classic analysis model — the worst-case analysis. In particular, we will introduce our worst-case analysis results on the *bandpass* problem from networks, and the *multiple RNA interaction* (MRIP) problem, the *maximum exemplar non-breakpoint similarity* (ENBS) problem and the *minimum integer partition* (MCIP) problem from bioinformatics. Then we will investigate the classic NP-hard problem, the *minimum independent dominating set* (MIDS) problem, under the smoothed analysis model. In order to look deep into the smoothed analysis model, we will finally show our smoothed analysis results on two classic data structures — *Trie* and *Patricia index tree*. All the results included in this thesis are from the collaborative papers to which I made significant contributions.

The *bandpass*-2 problem arises from optical communication networks using wavelength division multiplexing technology, and so it can be treated as a variant of the *maximum travelling salesman* problem. The difference between the *bandpass*-2 problem and the *maximum travelling salesman* problem is that, in former problem's setting, the edge weights are dynamic rather than fixed, which makes this problem much harder to solve than the *maximum travelling salesman* problem. We designed the first approximation algorithm with a performance ratio 19/36 [99], improving the previous best approximation ratio 1/2 [11, 64], dated back to 2004. Afterwards, another research group made a progress [28], but soon after we designed two more improved algorithms [53, 98] with some fascinating new combinatorial techniques on the *b-matching*. Our first algorithm and its theoretical analysis methods were presented on the *joint conference of the 6th*

International Frontiers of Algorithmics Workshop and the 8th International Conference on Algorithmic Aspects of Information and Management (FAW-AAIM 2012) [99]. One of the improved algorithms was submitted to the CoRR [98], and the other improved algorithm was accepted by the *Journal of Combinatorial Optimization* [53].

For the MRIP problem, RNA interactions are fundamental in many cellular processes, where two or multiple RNA molecules can be involved in the interactions. Multiple RNA interactions are believed much more complex than pairwise interactions. Recently, multiple RNA interaction prediction was formulated as a maximization problem. We extensively examined this optimization problem under several biologically meaningful interaction models. In particular, we presented a polynomial time algorithm for the problem when the order of interacting RNAs is known and *pseudoknot* interactions are allowed; for the general problem without an assumed RNA order, we proved the NP-hardness for both variants allowing and disallowing pseudoknot interactions, and presented a constant ratio approximation algorithm for each of them. These results were presented on the *7th Annual International Conference on Combinatorial Optimization and Applications (COCOA 2013)* [103], and the extended version was accepted by the *Theoretical Computer Science* [104].

For the ENBS problem, the genomic similarity measure, called *non-breakpoint similarity*, is the complement of the well-studied breakpoint distance between genomes (or in general, between any two sequences drawn from the same alphabet). For two genomes $\mathcal{G}$ and $\mathcal{H}$ drawn from the same set of n gene families and containing gene repetitions, we considered the corresponding ENBS problem, in which we deleted duplicated genes from $\mathcal{G}$ and $\mathcal{H}$ such that the resultant genomes G and H have the maximum non-breakpoint similarity. We obtained the following results. First, we proved that the *one-sided* 2-*repetitive* ENBS problem, *i.e.* when one of $\mathcal{G}$ and $\mathcal{H}$ is given exemplar and each gene occurs in the other genome at most twice, can be reduced from the *maximum independent set* problem with the instance size becomes squared. This implies that the ENBS problem does not admit any $O(n^{0.5-\epsilon})$-approximation algorithm, for any $\epsilon > 0$, unless NP = ZPP. This hardness result also implies that the ENBS problem is $W[1]$-hard. Secondly, we showed that the *two-sided* 2-*repetitive* ENBS problem has an $O(n^{0.5})$-approximation algorithm. These results were from the collaborative work of several research groups, which was published on the *Theoretical Computer Science* [25].

For the MCIP problem, we are given a collection of multisets $\{X_1, X_2, \ldots, X_k\}$ $(k \geq 2)$ of positive integers, a multiset S is a *common integer partition* (CIP) for them if S is an integer partition of every multiset $X_i, 1 \leq i \leq k$. The *minimum common integer partition* (k-MCIP) problem is defined as finding a CIP for $\{X_1, X_2, \ldots, X_k\}$ with the minimum cardinality. By some interesting combinatorial techniques, we presented a

$\frac{6}{5}$-approximation algorithm for the 2-MCIP problem, breaking an 8-year old record, as the previous best algorithm of ratio $\frac{5}{4}$ was designed in 2006 [21]. We then extended it to obtain a deterministic $3k/5$-approximation algorithm for the k-MCIP problem when k is even (when k is odd, the approximation ratio is $(3k+2)/5$). These results were presented on the *25th International Symposium on Algorithms and Computation (ISAAC 2014)* [105].

For the MIDS problem, it is well known that this problem does not admit a polynomial time approximation algorithm with worst-case performance ratio of $|V|^{1-\epsilon}$ for any $\epsilon > 0$ given the input graph $G = (V, E)$ [47]. We investigated it under the smoothed analysis model. In particular, we first studied the size of the minimum independent dominating set, denoted as $i(\mathfrak{g}(G,p))$, in perturbed graphs $\mathfrak{g}(G,p)$ and showed that $i(\mathfrak{g}(G,p))$ is *asymptotically almost surely* [1] in $\Theta(\log|V|)$. Furthermore, we proved that the probability of $i(\mathfrak{g}(G,p)) \geq \sqrt{4|V|/p}$ is no more than $2^{-|V|}$, and presented a simple greedy algorithm of worst-case performance ratio $\sqrt{4|V|/p}$ and with polynomial expected running time. These results were presented on the *19th Annual International Computing and Combinatorics Conference (COCOON 2013)* [100], and the extended version was accepted by the *Theoretical Computer Science* [101].

Trie and *Patracia index tree* are two classic data structures for storing strings. Let H_n denote the height of the Trie (the Patricia, respectively) on a set of n strings. It is well known that under the uniform distribution model on the strings, for Trie $H_n/\log n \to 2$ [36, 40, 41, 69, 80, 81, 83, 95, 96] and for Patricia $H_n/\log n \to 1$ [80], when n approaches infinity. Nevertheless, in the worst case, the height of the Trie on n strings is unbounded, and the height of the Patricia on n strings is in $\Theta(n)$. To better understand the practical performance of both the Trie and Patricia index trees, we investigated these two data structures in a smoothed analysis model. Given a set $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ of n binary strings, we perturb the set by adding an *i.i.d* Bernoulli random noise to each bit of every string. We showed that the resulting smoothed heights of Trie and Patricia trees are both $\Theta(\log n)$. These results were presented on the *20th Annual International Computing and Combinatorics Conference (COCOON 2014)* [102], and the extended version will be published on the *Theoretical Computer Science*.

1.2 Notations and preliminaries

In this section, some basic notations and concepts are introduced.

[1] In asymptotic analysis, one says that a property of the graph holds *asymptotically almost surely* (a.a.s.) if the property holds with the probability which converges to 1 as the size of the graph tends to ∞.

- **Family of Bachmann–Landau notations**:
  - $f(n) \in O(g(n))$: $\exists\ c > 0,\ \exists\ n_0,\ \forall\ n > n_0,\ f(n) \leq c \cdot g(n)$.
  - $f(n) \in \Omega(g(n))$: $\exists\ c > 0,\ \exists\ n_0,\ \forall\ n > n_0,\ f(n) \geq c \cdot g(n)$.
  - $f(n) \in \Theta(g(n))$: $\exists\ c_1 > 0,\ \exists\ c_2 > 0,\ \exists\ n_0,\ \forall\ n > n_0,\ c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$.
  - $f(n) \in o(g(n))$: $\forall\ c > 0,\ \exists\ n_0,\ \forall\ n > n_0,\ f(n) \leq c \cdot g(n)$.
  - $f(n) \in \omega(g(n))$: $\forall\ c > 0,\ \exists\ n_0,\ \forall\ n > n_0,\ f(n) \geq c \cdot g(n)$.
- **(one-tape) Turing machine**: A Turing machine can be formally defined as a 7-tuple $M = \langle Q, q_0, F, \Gamma, b, \Sigma, \delta \rangle$ where
  1. Q is a finite, non-empty set of states;
  2. $q_0 \in Q$ is the initial state;
  3. $F \subseteq Q$ is the set of final or accepting states;
  4. Γ is a finite, non-empty set of tape alphabet symbols;
  5. $b \in \Gamma$ is the blank symbol (the only symbol allowed to occur on the tape infinitely often at any step during the computation);
  6. $\Sigma \subseteq \Gamma \setminus \{b\}$ is the set of input symbols;
  7. $\delta : (Q \setminus F) \times \Gamma \nrightarrow Q \times \Gamma \times \{L, R\}$ is a partial function [2] called the *transition function*, where L is left shift, R is right shift.

  Anything that operates according to these specifications is a Turing machine. Roughly, a Turing machine can be imagined as a simple computer that reads and writes symbols one at a time on an endless tape by strictly following a set of rules.
- **Deterministic Turing machine (DTM)**: A deterministic Turing machine is a Turing machine whose set of rules prescribes at most one action to be performed for any given situation.
- **Non-deterministic Turing machine (NTM)**: A non-deterministic Turing machine is a Turing machine that may have a set of rules that prescribes more than one action for a given situation. For example, in a non-deterministic Turing machine, there may have both the following rules in its rule set.
  - If you are in state 2 and you see an 'A', change it to a 'B' and move right;
  - If you are in state 2 and you see an 'A', change it to a 'C' and move left.

[2] A *partial function* from X to Y (written as $f : X \nrightarrow Y$) is a function $f : X' \to Y$, for some subset X' of X. It generalizes the concept of a function $f : X \to Y$ by not forcing f to map every element of X to an element of Y (only some subset X' of X).

- **Probabilistic Turing machine**: A probabilistic Turing machine is a non-deterministic Turing machine which randomly chooses between the available transitions at each point according to some probability distribution.
- **Decision problem**: Decision problem is a special type of computational problem whose answer is either YES or NO.
- **P**: P is a class of decision problems that can be solved by a deterministic Turing machine using a polynomial amount of computation time. Following the convention, the time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the string representing the input.
- **NP**: NP is a class of decision problems for which a given solution can be verified in polynomial time by a deterministic Turing machine. An equivalent definition for NP is the following characterization: NP is the set of decision problems solvable by a non-deterministic Turing machine that runs in polynomial time.
- **NP-complete**: NP-complete is a class of problems which contains the hardest problems in NP. That is, every NP problem can be reduced to a NP-complete problem in polynomial time. Note that each element of NP-complete has to be an element of NP.
- **NP-hard**: NP-hard is a class of problems which are at least as hard as the hardest problems in NP. Problems in NP-hard do not have to be elements of NP, indeed, they may not even be decision problems.
- **Optimization problem**: Optimization problem is a class of problems of finding the best solution from all feasible solutions. Formally, a combinatorial optimization problem Π is a triple $(\mathcal{I}, \mathcal{F}, f)$, where $\mathcal{I}$ is a set of instances, $\mathcal{F}$ is the set of feasible solutions, f is the objective function mapping each feasible solution to some real (non-negative) value. The goal is to find a feasible solution such that its objective function is either minimized or maximized.
- **NP optimization problem (NPO)**: NP optimization problem is a class of combinatorial optimization problems with the following additional conditions:
  - the size of every feasible solution $s \in \mathcal{F}$ is polynomially bounded in the size of the given instance I;
  - the languages $\{I \mid I \in \mathcal{I}\}$ and $\{(I, s) \mid s \in \mathcal{F}\}$ [3] can be recognized in polynomial time;
  - $f(x)$ is polynomial-time computable.

[3] A *formal language* L over an alphabet Σ is a subset of Σ^*, that is, a set of words over that alphabet. Here these events are expressed into formal languages.

This implies that the corresponding decision problem is in NP.

- **ZPP**: ZPP is the complexity class of problems for which a probabilistic Turing machine exists with these properties:
  - it always returns the correct YES or NO answer.
  - the running time is polynomial in expectation for every input.
- **DTIME**: DTIME (or TIME) is the computational resource of computation time for a deterministic Turing machine. If a problem of input size n can require $f(n)$ computation time to solve, we have a complexity class DTIME($f(n)$) (or TIME($f(n)$)).

1.3 Approximation algorithms

With our current information technology development, there are an increasing number of optimization problems that need to be solved. An algorithm that solves a problem optimally while the time and space consumption are appropriately bounded is the ideal case. Unfortunately, many of the most interesting optimization problems are NP-hard, in other words, unless P = NP, we do not have efficient algorithms to compute the exact optimums for such problems. A classic NP-hard optimization problem is the *maximum independent set* problem: given a graph $G = (V, E)$, find a maximum size independent set $V_I \subset V$, where a subset of vertices is *independent* if no two vertices in the subset are connected by an edge.

One way to deal with these hard problems is to design algorithms by looking for trade-offs between the quality of the solution and the running time (or space consumption) of the algorithms. In this thesis, we address these hard problems by relaxing the requirement of finding an optimal solution. But we aim to compute a solution whose value is as close as possible to the value of the optimal solution. That is, we consider *approximation algorithms* for these hard optimization problems. We call a polynomial time algorithm $\mathcal{A}$ for an optimization problem Π a *ρ-approximation algorithm* if for all instances of the problem, $\mathcal{A}$ produces a solution whose value is within a factor of ρ of the value of an optimal solution. We would like to get ρ as close to 1 as possible.

More formally, consider an optimization problem $\Pi = (\mathcal{I}, \mathcal{F}, f)$, where $\mathcal{I}$ is a set of instances, $\mathcal{F}$ is the set of feasible solutions, f is the *objective function* mapping each feasible solution to some real (non-negative) value. Let $I \in \mathcal{I}$ be some instance of the problem Π. An *optimal solution* OPT(I) to I is the one that either maximizes or minimizes the value of the objective function f. And we call Π is a maximization or

minimization problem, respectively. Define the performance ratio of $\mathcal{A}$ on the instance I as

$$R_{\mathcal{A}} = \mathcal{A}(I)/\mathrm{OPT}(I). \tag{1.1}$$

Definition 1.1. (**Performance ratio** or **Performance factor**) A function $\rho : \mathbb{Z}^+ \to \mathbb{R}^+$ is called a *performance ratio* of the algorithm $\mathcal{A}$ if $R_{\mathcal{A}} \geq \rho(n)$ (or $R_{\mathcal{A}} \leq \rho(n)$) for all instances I of size n for the maximization (or minimization) problem Π. In this case, we refer to $\mathcal{A}$ as a $\rho(n)$-approximation algorithm if $\mathcal{A}$ is polynomial in the instance size n. In the special case where $\rho(n)$ is a constant, say ρ, we just say $\mathcal{A}$ is a ρ-approximation algorithm.

In this thesis, we will follow the convention that (1) the size of the instance is taken to be the size of the input in bits; (2) $\rho > 1$ for minimization problems while $\rho < 1$ for maximization problems. Since we want to obtain extremely good approximation algorithms such that the approximation ratio ρ is as close to 1 as possible, we are most interested in the *polynomial-time approximation schemes* (PTAS) and *fully polynomial-time approximation schemes* (FPTAS).

Definition 1.2. (**PTAS**) For any $\epsilon > 0$, suppose there is a $(1+\epsilon)$-approximation ($(1-\epsilon)$-approximation) algorithm $\mathcal{A}_\epsilon$ for the minimization (maximization) Π, such that $\mathcal{A}_\epsilon$'s running time is a polynomial of the instance size n. We call this family of algorithms as a *polynomial time approximation scheme* (PTAS) for Π.

Definition 1.3. (**FPTAS**) A PTAS is called a *fully polynomial time approximation scheme* (FPTAS) if the running time of $\mathcal{A}_\epsilon$ is a polynomial in $\frac{1}{\epsilon}$ and the instance size n.

Then we introduce three important concepts for the NP optimization problems.

Definition 1.4. (**APX**) APX is a class of NP optimization problems that allow polynomial-time approximation algorithms with approximation ratio bounded by a constant. The class APX is also sometimes known as **Max-SNP**.

Definition 1.5. (**APX-hard**) A problem is said to be *APX-hard* if there is a PTAS reduction [4] from every problem in APX to that problem.

Definition 1.6. (**APX-complete**) A problem is said to be *APX-complete* if the problem is APX-hard and also in APX.

There are many reasons forcing us to study the approximation algorithms. First of all, as we previously introduced, there are plenty optimization problems that hardly admit

[4] A *PTAS reduction* is an approximation-preserving reduction that preserves the property that a problem has a polynomial time approximation scheme (PTAS).

exact polynomial-time algorithms, for which approximation algorithms are increasingly being used. A typical example is the classic NP-hard problem — the *minimum vertex cover* problem. In the *minimum vertex cover* problem, given a graph $G = (V, E)$, we are required to find a vertex subset of the minimum cardinality such that each edge e in the given graph has at least one endpoint in this subset. Finding the optimum vertex cover is very time-consuming, which needs exponential running time even for planar graphs of degree at most 3 [44]. However, there is a simple and fast algorithm by keeping finding an uncovered edge and adding both endpoints to the vertex cover until no uncovered edges remain. It is clear that the running time is $O(|E|)$ and the resulting cover has size at most twice as large as the optimal one's, which implies this simple algorithm is a 2-approximation algorithm.

In some situations it is desirable to run an approximation algorithm even when there exists a polynomial-time algorithm for computing an exactly optimal solution. Because the approximation algorithm may have the benefit of faster running time, less space consumption, a much easier implementation, or it may lend itself more easily to a parallel or distributed implementation. These considerations become especially important when the input size is so astronomical that an exact polynomial-time algorithm with impractical running time, say $\Theta(n^{100000})$, would provide extremely bad performance in practice. Finally, studying approximation algorithms provides a mathematically rigorous basis on which to look deep into the problems, helps to figure out the problems' structures, and then may lead to a new algorithmic approach.

1.4 Three analysis methods

The analysis of an algorithm aims at providing measurement for the performance of the algorithm. The most commonly used theoretical approaches to understanding the behaviour of algorithms are the worst-case analysis and the average-case analysis. However, some of the well-known theoreticians, including Condon, Edelsbrunner, Emerson, Fortnow, Haber, Karp, Leivant, Lipton, Lynch, Parberry, Papadimitriou, Rabin, Rosenberg, Royer, Savage, Selman, Smith, Tardos, and Vitter, wrote (Challenges for Theory of Computing: Report of an NSF-Sponsored Workshop on Research in Theoretical Computer Science SIGACT News, 1999)

> *While theoretical work on models of computation and methods for analyzing algorithms has had enormous payoffs, we are not done. In many situations, simple algorithms do well. Take for example the Simplex algorithm for linear programming, or the success of simulated annealing on certain*

> *supposedly intractable problems. We don't understand why! It is apparent that worst-case analysis does not provide useful insights on the performance of many algorithms on real data. Our methods for measuring the performance of algorithms and heuristics and our models of computation need to be further developed and refined. Theoreticians are investing increasingly in careful experimental work leading to identification of important new questions in algorithms area. Developing means for predicting the performance of algorithms and heuristics on real data and on real computers is a grand challenge in algorithms.*

In this section, we will introduce the two classic analysis models, *i.e.* the worst-case and average-case analyses. Then a relatively new analysis model, named as the smoothed analysis, is also introduced, and a simple comparison is made among the three analysis models.

1.4.1 Worst-case analysis

The most common analysis toward understanding the performance of an algorithm is the worst-case analysis. The worst-case analysis requires to bound the worst possible performance an algorithm could achieve. In other words, this analysis is input independent and it provides a strong guarantee for the performance of an algorithm.

However, the worst-case analysis provides only one point of view on an algorithm's behaviour. In fact, under this point of view, the behaviour is often quite different from the typical behaviour, which the users are usually more interested in. Indeed, there are many computational or optimization problems in the real world, ranging from bioinformatics to social science, which can be solved by some heuristics or simple algorithms efficiently or effectively in most cases while these heuristics or algorithms have very poor worst-case performance either the running time or the approximation ratio taken as the performance measure.

A classic example is the simplex method, which is a kind of practical algorithms to solve linear programs and remains widely used today. Though almost all the simplex algorithms cost exponential time in the worst case, they often outperform many other polynomial time algorithms for linear programs, such as the ellipsoid algorithm [60] and the interior-point method [58], in the real applications [89–91]. Another example is the well-known algorithm Greedy for the *shortest common superstring* problem in bioinformatics. The *shortest common superstring* problem finds a shortest string s that contains every s_i as a substring for any given n strings $s_1, s_2, \ldots, s_n$. And it has been extensively

studied for its applications in string compression and DNA sequence assembly. There is a very simple algorithm GREEDY, which repeatedly merges two maximum overlapping strings into one until there is only one string left. This greedy algorithm works extremely well and it was reported that the average approximation ratio is below 1.014 for simulated data [84]. However, its approximation ratio under the worst-case analysis model is 3.5 [57].

1.4.2 Average-case analysis

To overcome the discrepancy between the poor worst-case performance and the practical performance, average-case analysis was introduced as an alternate. Under average-case analysis, a distribution of instances is first assumed and then the expected performance of the algorithm is measured. Ideally, we are given a mathematically analysable distribution which is the same or close to the real distribution. However, it is generally difficult to determine such a distribution because the distribution varies from area to area according to where the target algorithm is applied to. Furthermore, in most cases it is mathematically challenging to express the distribution using a small number of parameters. In most existing average-case analytical work, researchers have to use distributions with concise description, such as uniform distribution and Gaussian distribution, *etc.*, instead of the true but unknown distribution of the real-world instances. One can imagine that these commonly used special distributions may be far from the real ones and make the analysed instances bear very little resemblance to the real-world instances. In this sense, though a good average-case performance provides evidence that an algorithm may perform well in practice, it rarely fills up the gap between the practical world and theoretical world.

1.4.3 Smoothed analysis

Considering the drawbacks of both worst-case analysis and average-case analysis, Spielman and Teng [89] introduced the smoothed analysis to explain the performance of algorithms. The basic idea of smoothed analysis is to identify some typical properties of a given real-world instance, and can be regarded as a hybrid of worst-case analysis and average-case analysis. More formally, it measures the worst-case expected performance of an algorithm under slight random perturbation of an instance.

We next see how this analysis method relates to the worst-case and the average-case analysis methods. In the following content, to distinguish a matrix or a vector from a scalar, let notations with bold font represent matrices or vectors, if there is no extra

explanation. Besides, a constant with bold font means a matrix or a vector with each component equal to this constant, say $\mathbf{1}$ denotes a matrix or a vector consisting of 1's.

Formally, let Q be a quality measurement, and without loss of generality, assume Q has a property that the larger the worse, such as the running time. Suppose $\mathcal{A}$ is the algorithm we want to analyse, $\boldsymbol{x}$ is (the string representation of) an input instance, $Q(\mathcal{A}, \boldsymbol{x})$ is the instance based quality measurement, D is the universe of instances, D_n is the subset of all size n instances in D. The smoothed measure of $\mathcal{A}$ under Q is

$$\mathcal{Q}_{smooth}(n, \sigma) = \sup_{\bar{\boldsymbol{x}} \in D_n} E_{\boldsymbol{r}}\{Q(\mathcal{A}, \bar{\boldsymbol{x}} + \sigma \cdot \boldsymbol{r})\},$$

where $\sigma \in \mathbb{R}^+$, called the *perturbation parameter*, measures the strength of noise, and $\boldsymbol{r}$ is some noise vector, which has the same dimension as $\bar{\boldsymbol{x}}$'s.

Then we have the following observations. When the perturbation parameter σ is extremely small, $\bar{\boldsymbol{x}} + \sigma \cdot \boldsymbol{r} \approx \bar{\boldsymbol{x}}$, which means the smoothed analysis becomes the worst-case analysis; when σ becomes larger, the perturbed instances would have more randomness and extremely the smoothed analysis would become the average-case analysis. Therefore, by varying the perturbation parameter σ, the smoothed analysis interpolates between these two extreme cases. Usually we are more interested in the case where σ is relatively small, because real-world instances are often subject to a slight amount of noise. For example, when input parameters are obtained from physical measurements of real-world phenomena, the measurements usually have some random uncertainty of low magnitudes; besides, if the input parameters are the output of some computer programs, the numerical calculation of computer programs may also add some uncertainty due to the numerical imprecision. An example for noise in discrete application is that building a complicate transportation network is governed by some blueprint of the government or contractor but the blueprint may still be "perturbed" due to some unpredictable uncertainty, such as fluctuation of funding budget, some nail household that refuses uncompromisingly to move when the land is requisitioned for the construction project, *etc.*.

An algorithm with a good worst-case analysis will perform well on all instances, as the worst possible performance of the algorithm is bounded under this analysis model. If the smoothed measure of $\mathcal{A}$ under Q is good with some relatively small σ and some reasonable random model for $\boldsymbol{r}$, the hard instances are "isolated" in the instance space and it is unlikely the measure of $\mathcal{A}$ under Q will be every bad in real world application.

1.4.3.1 Approximability smoothed analysis

Spielman and Teng [89] first introduced the smoothed analysis to explain the success of the simplex algorithm with the shadow-vertex pivoting rule in real applications. They considered the running time as the performance measure and proposed the concept of *polynomial smoothed complexity*.

Definition 1.7 (**Polynomial Smoothed Complexity [89]**)**.** Given a problem Π with input domain $D = \cup_n D_n$ where D_n represents all size n instances. Let $\mathcal{R} = \cup_{n,\sigma} R_{n,\sigma}$ be a family of perturbations where $R_{n,\sigma}$ defines for each $\bar{\boldsymbol{x}} \in D_n$ a perturbation distribution of $\bar{\boldsymbol{x}}$ with magnitude σ. Let $\mathcal{A}$ be an algorithm for solving Π and $T_{\mathcal{A}}(\boldsymbol{x})$ be the time for solving an instance $\boldsymbol{x} \in D$. Then algorithm $\mathcal{A}$ has polynomial smoothed complexity if there exist constants n_0, σ_0, c, k_1 and k_2 such that for all $n > n_0$ and $0 \leq \sigma \leq \sigma_0$,

$$\max_{\bar{\boldsymbol{x}} \in D_n} \{E_{\boldsymbol{x} \sim R_{n,\sigma}(\bar{\boldsymbol{x}})}[T_{\mathcal{A}}(\boldsymbol{x})]\} \leq c \cdot \sigma^{-k_1} \cdot n^{k_2},$$

where $\boldsymbol{x} \sim R_{n,\sigma}(\bar{\boldsymbol{x}})$ means $\boldsymbol{x}$ follows the distribution $R_{n,\sigma}(\bar{\boldsymbol{x}})$.

The problem Π has smoothed polynomial time complexity with some perturbation model $\mathcal{R}$ if it admits an algorithm with a polynomial smoothed complexity.

Spielman and Teng [89] proved that the simplex algorithm with the shadow-vertex pivoting rule is smoothed polynomial under the Gaussian perturbation, that is, the maximum over $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{y}}$ of expected running time of the above simplex algorithm on the following inputs is bounded by a polynomial in m, d, σ.

$$\begin{aligned} \text{Maximize} \quad & \boldsymbol{z}^T \boldsymbol{x} \\ \text{subject to} \quad & (\bar{\boldsymbol{A}} + \boldsymbol{G}) \cdot \boldsymbol{x} \leq (\bar{\boldsymbol{y}} + \boldsymbol{h}), \end{aligned}$$

where $\bar{\boldsymbol{A}}_{m \times d}$ and $\bar{\boldsymbol{y}}_{m \times 1}$ are arbitrary given matrix and vector respectively; $\boldsymbol{G}$ and $\boldsymbol{h}$ are a matrix and a vector, respectively, consisting of independent Gaussian random variables of mean $\mathbf{0}$ and standard deviation $\boldsymbol{\sigma'}$ with $\sigma' = \sigma \cdot \max_i ||(\bar{y}_i, \bar{\boldsymbol{a}}_{\boldsymbol{i}})||$, here $\bar{\boldsymbol{a}}_{\boldsymbol{i}}$ is the ith row of $\bar{\boldsymbol{A}}$.

Since then, smoothed analysis has been applied successfully to a variety of different algorithms and problems: mathematical programming, scientific computing, game theory, graph theory, AI related problems and discrete combinatorial optimization problems *etc.*. For more detailed surveys of the smoothed analysis, one may refer to [13, 68, 90–92]. Originally, smoothed analysis was introduced to evaluate the performance of an algorithm by its running time. There are many other performance measures that are also very important and reveal some typical properties of an algorithm. For example,

the amount of storage occupied during execution of the algorithm, the number of bits of precision required to achieve a given output accuracy, the number of cache misses, the error probability of a decision problem, the number of random bits needed in a randomized algorithm, the number of calls to a particular subroutine, the number of iterations of an iterative algorithm [92].

For an approximation algorithm, we usually concentrate on the quality of the solution it returns, that is, how well the solution could approximate the optimal one. Therefore the approximation ratio is regarded as the major performance measure instead of the running time because an approximation algorithm has polynomial time complexity under the worst-case analysis model and time complexity based analysis seems to be less of interest in some sense. In most of the existing work, the performance measures of the algorithms are usually running time or space consumes. On the other hand, in real world applications, lots of approximation algorithms perform very well in practice but have poor approximation ratios under the worst-case analysis, such as the Greedy for the *shortest common superstring* problem as introduced in Section 1.4.1. Therefore, smoothed analysis on the performance ratio of approximation algorithms would help us to understand these algorithms better. Besides, for a certain problem, there may exist some quantity that reveals some essential properties of the problem itself, which in turn may help us to better understand the problem or to design new more efficient and/or more effective algorithms. Thus measuring such quantities under a reasonable model, say smoothed analysis, would be of great significance.

1.4.3.2 Approximability based algorithm design

Smoothed analysis helps us to understand the behaviour of an algorithm better by revealing its typical properties. Moreover, it helps us to look deep into the problem itself. In turn, the insight gained from the smoothed analysis results may inspire us with new ideas in algorithm design for real applications.

For example, inspired by the smoothed analysis for simplex method [35, 89, 106], Kelner and Spielman [59] proposed the first randomized polynomial-time simplex algorithm for linear programs. Here is another simple example. Sankar [86] first investigated the Gaussian Elimination by smoothed analysis. This idea was exploited by Spielman and Teng [92], who suggested a more stable solver for linear systems. Suppose we are given a linear system $\boldsymbol{Ax} = \boldsymbol{b}$ and with error tolerance δ. Consider the following algorithm [92].

1. Use the standard Gaussian Elimination with partial pivoting rule [5] to solve $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Suppose $\boldsymbol{x}^*$ is the solution returned.

2. If $||\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^*|| < \delta$, return $\boldsymbol{x}^*$.

3. Otherwise, add a small noise and generate a new linear system $(\boldsymbol{A} + \epsilon \cdot \boldsymbol{G})\boldsymbol{y} = \boldsymbol{b}$, where ϵ is a small positive number and $\boldsymbol{G}$ is a Gaussian matrix with mean $\boldsymbol{0}$ and variance $\boldsymbol{1}$.

4. Solve the perturbed linear system with Gaussian Elimination without pivoting and return the solution.

Sankar [86] proved that if ϵ is sufficiently smaller than $\boldsymbol{A}$'s condition number $\kappa(\boldsymbol{A})$, the solution to the perturbed linear system can well approximate the original one. He also proved that the quality of the growth factor cannot be too large with high probability. Thus, the Gaussian Elimination with partial pivoting on the original linear system may fail due to the large growth factor, but the success of the new algorithm only depends on the machine precision and condition number of $\boldsymbol{A}$. [6]

Recently, this smoothed analysis based algorithm design method was generalized from the time complexity smoothed analysis to approximability smoothed analysis by Manthey and Plociennik [67]. After studying independent number under the smoothed analysis with a p-Boolean perturbation model (as introduced in Section 1.4.3.3), they presented an algorithm approximating the independence number $\alpha(\mathfrak{g}(G,p))$ with a worst-case approximation ratio $O(\sqrt{|V| \cdot p})$ and with polynomial expected running time for sufficiently large p, where $G = (V, E)$ is the given graph, $\mathfrak{g}(G,p)$ is the perturbed graph of G [7] and the independent number is the size of the maximum independent set.

1.4.3.3 Perturbation models

As we discussed above, the perturbation model would be very important to the whole smoothed analysis model. The perturbation model that captures the randomness and imprecision of the input parameters can vary from application to application. The following perturbation models are commonly used in existing work on the smoothed analysis.

Continuous perturbations:

[5] In partial pivoting, the algorithm selects the entry with largest absolute value from the column of the matrix that is currently being considered as the pivot element.

[6] There is a simple Matlab experiment for this new algorithm in [92].

[7] The *perturbed graph* of the input graph $G = (V, E)$ is obtained by negating the existence of edges independently with a probability $p > 0$.

Definition 1.8 (**Gaussian Perturbation).** Let $\bar{\boldsymbol{x}} \in D_n$. A σ-Gaussian perturbation of $\bar{\boldsymbol{x}}$ is $\boldsymbol{x} = \bar{\boldsymbol{x}} + \boldsymbol{g}$, where $\boldsymbol{g}$ is a Gaussian random vector of mean $\mathbf{0}$ and variance $\boldsymbol{\sigma^2}$ with $\sigma \in \mathbb{R}^+$.

Definition 1.9 (**Relative Gaussian Perturbation).** Let $\bar{\boldsymbol{x}} \in D_n$. A relative σ-Gaussian perturbation of $\bar{\boldsymbol{x}}$ is $\boldsymbol{x} = \bar{\boldsymbol{x}} \cdot (1 + g)$, where g is a Gaussian random variable of mean 0 and variance σ^2 with $\sigma \in \mathbb{R}^+$.

Definition 1.10 (**Uniform Ball Perturbation).** Let $\bar{\boldsymbol{x}} \in D_n$. A uniform ball perturbation of radius σ of $\bar{\boldsymbol{x}}$ is a random vector $\boldsymbol{x}$ drawn uniformly from the ball of radius σ centered at $\bar{\boldsymbol{x}}$.

Discrete perturbations:

Definition 1.11 (**Boolean Perturbation).** Let $\bar{\boldsymbol{x}} = (\bar{x}_1, \cdots, \bar{x}_n) \in \{0,1\}^n$ or $\{-1,1\}^n$. A p-Boolean perturbation of $\bar{\boldsymbol{x}}$ is a random string $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ or $\{-1,1\}^n$, with $\Pr\{x_i = \bar{x}_i\} = 1 - p$.

Definition 1.12 (**Partial Bit Randomization).** Let $\bar{x}$ be an integer with K-bit binary representation and $k \leq K$ be a positive integer. A k-partial bit randomization of $\bar{x}$ is an integer x obtained by replacing $\bar{x}$'s k least significant bits (also referred to as the right-most bits) by the binary expression of a random integer from $[0, 2^{k-1}]$ according to some specific distribution, say uniform distribution, over $[0, 2^{k-1}] \cap \mathbb{Z}$.

Definition 1.13 (**Partial Permutation).** Let $\bar{s}$ be a sequence of n elements and $p \in [0,1]$. A p-partial permutation of $\bar{s}$ is a random sequence s by first creating a set S by selecting independently with $\Pr\{i \in S\} = p, i = 1, \ldots, n$, and then uniformly permuting elements of $\bar{s}$ in position S while all other elements remaining unchanged.

In the above perturbation models, inputs are perturbed at random, which may be unnatural for some problems. Thus, it might be necessary to add some constraints on the perturbation by requiring that inputs should have some typical properties. For example, a person's body temperature can neither be too high nor too low, and we should not allow perturbations of the temperature of human body that violate this constraint to enter our probability space. In general, the perturbations should make sure that any perturbed instance carries some certain significant aspects. Spielman and Teng [90] proposed the concept named *property-preserving perturbation*, which is defined by restricting a natural perturbation model to preserve certain properties of the original input. More details about the *property-preserving perturbation* model can be found in the survey by Spielman and Teng [90].

Here, we need to mention another very powerful smoothed analysis model, named as *one-step model*, due to Beier and Vöcking [10]. In this model, an adversary is allowed

to specify the probability density function for each input value in the input instance. To prevent the adversary from modelling a worst-case instance too closely, we bound the density functions from above by a *smoothing parameter* ϕ. Roughly speaking, a large ϕ forces the algorithm to perform almost as bad as on worst-case instances while a relatively small ϕ makes the adversary to choose the uniform distribution on the input space, which mimics an average-case analysis. For example, suppose each input instance of some problem contains a value vector $\boldsymbol{v} = (v_1, \ldots, v_n), v_i \in [0, 1], i \in \{1, \ldots, n\}$. The adversary does not fix the value of each v_i, instead he specifies probability density functions $f_i : [0, 1] \to [0, \phi]$ according to which the value v_i are randomly drawn independently of each other. If $\phi = 1$, then the adversary has no choice but to specify a uniform distribution on the interval $[0, 1]$ for each value v_i. In this case, our analysis becomes an average-case analysis. On the other hand, if ϕ becomes large, then the analysis approaches a worst-case analysis since the adversary can specify small interval I_i of length $\frac{1}{\phi}$ (that contains the values in a worst-case instance) for each value v_i from which the value v_i is drawn uniformly. Thus, the adversarial *smoothing parameter* ϕ serves as an interpolation parameter between the worst-case and average-case analyses.

Chapter 2

Bandpass Problem[1]

2.1 Introduction

The *bandpass*-2 problem is a variant of the *maximum travelling salesman* problem arising from optical communication networks. In optical communication networks, a sending point uses a binary matrix $A_{n\times m}$ to send n information packages to m different destination points, in which the entry $a_{ij} = 1$ if information package i is *not* destined for point j, or $a_{ij} = 0$ otherwise. To achieve the highest cost reduction via wavelength division multiplexing technology, an optimal packing of information flows on different wavelengths into groups is necessary [8]. Under this binary matrix representation, every B consecutive 1's in a column indicates an opportunity for merging information to reduce the communication cost, where B is a pre-specified positive integer called the *bandpass number*. Such a set of B consecutive 1's in a column of the matrix is said to form a *bandpass*. When counting the number of bandpasses in the matrix, no two of them in the same column are allowed to share any common rows. The computational problem, the *bandpass-B* problem, is to find an optimal permutation of rows of the input matrix $A_{n\times m}$ such that the total number of extracted bandpasses in the resultant matrix is maximized [9, 11, 64]. Note that though multiple bandpass numbers can be used in practice, for the sake of complexities and costs, usually only one fixed bandpass number is considered [9].

The general *bandpass-B* problem, for any fixed $B \geq 2$, has been proven to be NP-hard [64]. In fact, the NP-hardness of the *bandpass*-2 problem can be proven by a reduction from the well-known *Hamiltonian path* problem [44], where in the constructed binary matrix $A_{n\times m}$, a row maps to a vertex, a column maps to an edge, and $a_{ij} = 1$ if and only if edge e_j is incident to vertex v_i. It follows that there is a row permutation achieving $n - 1$ bandpasses if and only if there is a Hamiltonian path in the graph.

On the approximability, the *bandpass-B* problem has a close connection to the *weighted B-set packing* problem [44]. By taking advantages of the approximation algorithms designed for the *weighted B-set packing* problem [8, 18], the *bandpass-B* problem can be

[1] This chapter is based on [53, 98, 99].

approximated within $O(B^2)$ [64]. Moreover, since the *maximum weight matching* problem is solvable in cubic time, the *bandpass*-2 problem admits a simple maximum weight matching based 2-approximation algorithm [64]. In the sequel, we call the *bandpass*-2 problem simply the *bandpass* problem.

Recently in 2012, Tong *et al.* [99] first presented an intrinsic structural property for the optimal row permutation. That is, with respect to a maximum weight matching between the rows, the *isolated* bandpasses (meaning a 0 above and a 0 below each such bandpass) in the optimal row permutation can be classified into four disjoint groups; after extracting bandpasses of the matching out of the instance, certain fractions of these four groups of *isolated* bandpasses remain in the residual instance. They proposed to compute another maximum weight matching in the residual instance, and from which to extract a sub-matching to extend the first maximum weight matching into an acyclic 2-matching. The acyclic 2-matching is then formed into a row permutation, which leads to an approximation algorithm with a worst-case performance ratio of $19/36 \approx 0.5277$.

As we can see, this performance analysis is essentially based on breaking cycles that are formed in the union of two matchings [99]; this can be equivalently deemed as partitioning (the edge set of) the second matching into two sub-matchings, such that the union of each of them and the first maximum weight matching is acyclic. Subsequently, Chen and Wang [28] presented an alternative to compute the target sub-matching (to extend the first maximum weight matching into an acyclic 2-matching). They showed that a maximum weight 2-matching (acyclic 2-matching, respectively) can be partitioned into 4 (3, respectively) candidate sub-matchings. Using the best of these candidate sub-matchings in their algorithm to extend the first maximum weight matching guarantees a solution row permutation that contains at least a fraction $117/220 \approx 0.5318$ of the bandpasses in the optimum [28]. Soon afterwards, Tong *et al.* [98] proposed an improvement to compute a maximum weight 4-matching in the residual instance, and to show how to partition it into 7.5 candidate sub-matchings. The improved approximation algorithm has a worst-case performance ratio of $227/426 \approx 0.5328$.

Later, Tong *et al.* [53] presented another novel scheme to partition a 4-matching into a number of candidate sub-matchings, each of which can be used to extend the first maximum weight matching into an acyclic 2-matching. They showed that among these sub-matchings the maximum weight can be guaranteed to a better extent, and thus proved a new approximation algorithm of worst-case performance ratio $\frac{70-\sqrt{2}}{128} \approx 0.5358$. At the end, Tong *et al.* [53] concluded that this 0.5358-approximation algorithm seems to have taken full advantage of the structural property of the optimal row permutation.

In the next section, we introduce some basic concepts and some important lemmas. In Section 2.3, 2.4, 2.5, we will introduce three main algorithms for the *bandpass* problem, denoted as BP1, BP2, BP3, respectively.

2.2 Preliminary

The NP-hardness of the *bandpass* problem is confirmed via a reduction from the *Hamiltonian path* problem [64]. However, the *bandpass* problem does not readily reduce to the *maximum traveling salesman* (Max-TSP) problem [44] for the approximation algorithm design. As pointed out in [98], an instance graph of Max-TSP is *fixed*, in that all (non-negative) edge weights are given at the beginning, while in the *bandpass* problem the number of bandpasses extracted between two consecutive rows in a row permutation is permutation dependent. Nevertheless, as shown in the sequel, the algorithm design is still based on maximum weight b-matchings for $b = 1, 2, 4$, similarly as in approximating the Max-TSP [27, 50, 78, 88]. Formally, in the Max-TSP problem, a complete edge-weighted graph is given, where the edge weights are non-negative integers, and the goal is to compute a Hamiltonian cycle with the maximum weight. Note that there are several variants of the Max-TSP problem studied in the literature. In our case, the input graph is undirected (or symmetric) and the edge weights do not necessarily satisfy the triangle inequality.

In our *bandpass* problem, since we can always add a row of all 0's if necessary, we assume, without loss of generality, that the number of rows, n, is even. A *b-matching* of a graph is a subgraph in which the degree of each vertex is at most b. A maximum weight b-matching of an edge weighted graph can be computed in $O(n^2m)$ time [7, 43, 71], where n is the number of vertices and m is the number of edges in the graph. Note that a 2-matching is a collection of vertex-disjoint cycles and paths. A 2-matching is *acyclic* if it does not contain any cycle (*i.e.*, it is a collection of vertex-disjoint paths). A matching M *extends* another matching M' into an acyclic 2-matching if and only if the union of these two matchings is acyclic.

Given the input binary matrix $A_{n\times m}$, let r_i denote the i-th row. We first construct a graph G of which the vertex set is exactly the row set $\{r_1, r_2, \ldots, r_n\}$. Between rows r_i and r_j, the *fixed* edge weight is defined as the maximum number of bandpasses that can be formed between the two rows (*i.e.*, the number of columns both rows have 1 in) and is denoted as $w(i, j)$. In the sequel we use row (of the matrix) and vertex (of the graph) interchangeably.

For a row permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$, its i-th row is the π_i-th row in the input matrix. We call a maximal segment of consecutive 1's in a column of π a *strip* of π. The length of a strip is defined to be the number of 1's therein. A length-ℓ strip contributes exactly $\left\lfloor \frac{\ell}{2} \right\rfloor$ bandpasses to the permutation π. We use $S_\ell(\pi)$ to denote the set of all length-ℓ strips of π, and $s_\ell(\pi) = |S_\ell(\pi)|$. Let $b(\pi)$ denote the number of bandpasses extracted from the permutation π, and $p(\pi)$ denote the number of pairs of consecutive 1's in the permutation π. Notice that a length-ℓ strip contributes exactly $\ell - 1$ pairs to the permutation π. Based on the previous definition, we have the following two equations.

$$b(\pi) = \sum_{\ell=2}^{n} s_\ell(\pi) \left\lfloor \frac{\ell}{2} \right\rfloor = s_2(\pi) + \sum_{\ell=3}^{n} s_\ell(\pi) \left\lfloor \frac{\ell}{2} \right\rfloor, \tag{2.1}$$

$$p(\pi) = \sum_{\ell=2}^{n} s_\ell(\pi)(\ell - 1) = s_2(\pi) + \sum_{\ell=3}^{n} s_\ell(\pi)(\ell - 1). \tag{2.2}$$

2.2.1 Algorithm template

For the three approximation algorithms BP1, BP2 and BP3, the rough ideas are the same. The first step is to compute a maximum weight matching M_1 in graph G. Recall that there are an even number of rows. Therefore, M_1 is a perfect matching (even though some edge weights could be 0). Let $w(M_1)$ denote the sum of its edge weights, indicating that exactly $w(M_1)$ bandpasses can be extracted from the row pairings suggested by M_1. These bandpasses are called the bandpasses of M_1.

Next, every 1 involved in a bandpass of M_1 is changed to 0. Let the resultant matrix be denoted as $A'_{m\times n}$, the resultant edge weight between rows r_i and r_j be $w'(i, j)$ — which is the maximum number of bandpasses that can be formed between the two revised rows — and the corresponding resultant graph be denoted as G'. One can see that if an edge (r_i, r_j) belongs to M_1, then the new edge weight $w'(i, j) = 0$. In the second step, we compute a matching M_2 in graph G'. Let $w'(M_2)$ denote its weight or its number of bandpasses. It is noted that no bandpass of M_1 shares a 1 with any bandpass of M_2.

The last step is based on whether $G[M_1 \cup M_2]$ is acyclic or not. If $G[M_1 \cup M_2]$ is cyclic, we need to break cycles first, by removing for each cycle the least weight edge of M_2. Otherwise, we do nothing. Finally we stack these paths arbitrarily in the remaining acyclic graph to give a row permutation π. It is not hard to see that the number of bandpasses extracted from π is $w(M_1) + \frac{1}{2}w'(M_2)$ and $w(M_1) + w'(M_2)$, respectively for the case $G[M_1 \cup M_2]$ is acyclic and acyclic, respectively. The main differences and the beauty of the algorithms lie in how we calculate the second matching M_2.

2.2.2 Key structure of bandpass

Let π^* denote the optimal row permutation such that its $b(\pi^*)$ is maximized over all row permutations. Correspondingly, $S_2(\pi^*)$ denotes the set of length-2 strips in π^*, which contributes exactly $s_2(\pi^*)$ bandpasses towards $b(\pi^*)$. The key part in the performance analysis for the algorithms BP1, BP2, BP3 is to estimate $w'(M_2)$, as done in the following.

First, we partition the bandpasses of $S_2(\pi^*)$ into four groups: B_1, B_2, B_3, B_4. Note that bandpasses of $S_2(\pi^*)$ do not share any 1 with each other. B_1 consists of the bandpasses of $S_2(\pi^*)$ that also belong to matching M_1 (such as the one between rows r_a and r_b in FIGURE 2.1); B_2 consists of the bandpasses of $S_2(\pi^*)$, each of which shares (exactly) a 1 with exactly one bandpass of M_1, and the other 1 of the involved bandpass of M_1 is shared by another bandpass in B_2; B_3 consists of the bandpasses of $S_2(\pi^*)$, each of which shares (exactly) a 1 with at least one bandpass of M_1, and if it shares a 1 with exactly one bandpass of M_1 then the other 1 of the involved bandpass of M_1 is not shared by any other bandpass of B_2; B_4 consists of the remaining bandpasses of $S_2(\pi^*)$. FIGURE 2.1 illustrates some examples of these bandpasses.

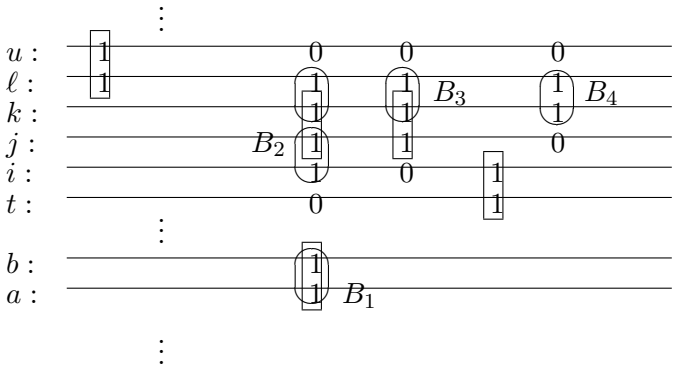

FIGURE 2.1: An illustration of the bandpasses of $S_2(\pi^*)$ (in ovals) and the bandpasses of M_1 (in boxes) for grouping purpose. A horizontal line in the figure represents a row, led by its index. Rows that are adjacent in π^* and/or row pairs of M_1 are intentionally ordered adjacently. In this figure, rows r_a and r_b are adjacent in π^*, denoted as $(r_a, r_b) \in \pi^*$, and edge $(r_a, r_b) \in M_1$ as well; the bandpasses between these two rows in $S_2(\pi^*)$ thus belong to B_1. Edges $(r_t, r_i), (r_j, r_k), (r_\ell, r_u) \in M_1$, while $(r_i, r_j), (r_k, r_\ell) \in \pi^*$; the bandpasses between rows r_i and r_j and between rows r_k and r_ℓ in $S_2(\pi^*)$ shown in the figure have their group memberships indicated beside them respectively.

By the definition of partition, we have

$$s_2(\pi^*) = |B_1| + |B_2| + |B_3| + |B_4|. \tag{2.3}$$

From these "group" definitions, we know all bandpasses of B_1 are in M_1. Also, one pair of bandpasses of B_2 correspond to a distinct bandpass of M_1. Bandpasses of B_3 can be further partitioned into subgroups such that a subgroup of bandpasses together with a distinct maximal subset of bandpasses of M_1 form into an alternating cycle or path of length at least 2. Moreover, 1) when the path length is even, the number of bandpasses of this subgroup of B_3 is equal to the number of bandpasses of this subset of bandpasses of M_1; 2) when the path length is odd, 2a) either the number of bandpasses of this subgroup of B_3 is 1 greater than the number of bandpasses of this subset of bandpasses of M_1, 2b) or the path length has to be at least 5 and so the number of bandpasses of this subgroup of B_3 is at least $\frac{2}{3}$ of the number of bandpasses of this subset of bandpasses of M_1. It follows from 1), 2a) and 2b) that with respect to B_3, M_1 contains at least $\frac{2}{3}|B_3|$ corresponding bandpasses. That is,

$$w(M_1) \geq |B_1| + \frac{1}{2}|B_2| + \frac{2}{3}|B_3|. \tag{2.4}$$

Apparently, all bandpasses of B_4 are in graph G', while none of $B_1 \cup B_2 \cup B_3$ is in graph G'.

Note that the bandpasses of B_2 are paired up such that each pair of the two bandpasses share a 1 with a bandpass of M_1. Assume without loss of generality that these two bandpasses of B_2 are formed between rows r_i and r_j and between rows r_k and r_ℓ, respectively, and that the involved bandpass of M_1 is formed between rows r_j and r_k (see FIGURE 2.1). That is, in the optimal row permutation π^*, rows r_i and r_j are adjacent, and rows r_k and r_ℓ are adjacent; while edge $(r_j, r_k) \in M_1$. We remark that these four rows are distinct. We conclude that edge $(r_i, r_\ell) \notin M_1$. The proof is simple as otherwise in the particular column a bandpass would be formed between rows r_i and r_ℓ, making the two bandpasses of B_2 lose their group memberships (*i.e.*, they would belong to B_3).

Lemma 2.1. *Assume edge $(r_j, r_k) \in M_1$, and that one bandpass of (r_j, r_k) shares 1 with (two) bandpasses of B_2. Then in G edge (r_j, r_k) is adjacent to at most four edges in the optimal row permutation π^*, at most two of which are incident at row r_j and at most two of which are incident at row r_k.*

Proof. The lemma is straightforward from the above discussion, and the fact that edge (r_j, r_k) does not belong to π^*. □

Continuing with the above discussion, assuming that edge $(r_j, r_k) \in M_1$, and that one bandpass of (r_j, r_k) shares 1 with two bandpasses of B_2, which are formed between rows r_i and r_j and between rows r_k and r_ℓ, respectively (see FIGURE 2.1). We know that

in graph G', between rows r_i and r_ℓ, in the same column there is a bandpass (which contributes 1 towards the edge weight $w'(i,\ell)$). We call bandpasses constructed in this way the *induced* bandpasses. From Lemma 2.1, edge (r_j, r_k) is adjacent to at most two edges of π^* incident at row r_j. It follows that in graph G', row r_ℓ can form induced bandpasses with at most four other rows. In the other words, the subgraph of G' induced by the edges containing induced bandpasses, denoted as G'_s, is a degree-4 graph.

Lemma 2.2. *G'_s is a degree-4 graph, and its weight $w'(G'_s) \geq \frac{1}{2}|B_2|$.*

Proof. The first half of the lemma is a result of the above discussion. Since every pair of bandpasses of B_2 leads to an induced bandpass, all the edge weights in G'_s sum up to at least $\frac{1}{2}|B_2|$, which is the number of bandpass pairs in B_2. □

2.3 $\frac{19}{36}$-approximation algorithm BP1

In this section, we will introduce our first improved approximation algorithm BP1 for the *bandpass* problem.

2.3.1 Algorithm description

As introduced in the Section 2.2, the first step of BP1 is as same as the algorithm template described in Section 2.2.1. In the second step of BP1, we compute a maximum weight matching M_2 in graph G'. If an edge (r_i, r_j) belongs to both M_1 and M_2, then it is removed from M_2. Such a removal does not decrease the weight of M_2 as $w'(i,j) = 0$. Consider the union of M_1 and M_2, denoted as $G[M_1 \cup M_2]$. Note that every cycle of this union, if any, must be an even cycle with alternating edges of M_1 and M_2. The third step of BP1 is to break cycles, by removing for each cycle the least weight edge of M_2. Let M denote the final set of edges of the union, which form into disjoint paths. In the last step, we arbitrarily stack these paths to give a row permutation π. The number of bandpasses extracted from π, $b(\pi)$, is at least the weight of M, which is greater than or equal to $w(M_1) + \frac{1}{2}w'(M_2)$.

2.3.2 Performance analysis

Lemma 2.3. *The weight of matching M_2 is $w'(M_2) \geq \max\{\frac{1}{10}|B_2|, \frac{1}{2}|B_4|\} \geq x\frac{1}{10}|B_2| + (1-x)\frac{1}{2}|B_4|$, for any $x \in [0,1]$.*

Proof. Vizing's Theorem [107] states that the edge coloring (chromatic) number of a graph is either the maximum degree Δ or $\Delta + 1$. Note that all edges of the same color form a matching in the graph. We conclude from Lemma 2.2 that, even in graph G'_s there is a matching of weight at least $\frac{1}{5}w'(G'_s) \geq \frac{1}{10}|B_2|$. As G'_s is a subgraph of G' and M_2 is the maximum weight matching of G', $w'(M_2) \geq \frac{1}{10}|B_2|$.

On the other hand, graph G' contains all bandpasses of B_4. Therefore, $w'(M_2) \geq \frac{1}{2}|B_4|$ as well. The last inequality in the lemma then follows trivially,

$$\max\left\{\frac{1}{10}|B_2|, \frac{1}{2}|B_4|\right\} \geq x\frac{1}{10}|B_2| + (1-x)\frac{1}{2}|B_4|,$$

for any $x \in [0,1]$. □

Theorem 2.4. *Algorithm* BP1 *is a cubic time* $\frac{19}{36}$*-approximation for the bandpass problem.*

Proof. The running time of algorithm BP1 is dominated by the computing for two maximum weight matchings, which can be done in cubic time. Since M_1 is the maximum weight matching in graph G, from Eq. (2.2) we have

$$w(M_1) \geq \frac{1}{2}p(\pi^*) \geq \frac{1}{2}\left(s_2(\pi^*) + \sum_{\ell=3}^{m} s_\ell(\pi^*)(\ell-1)\right). \tag{2.5}$$

Combining Eqs. (2.4) and (2.5), we have for any $y \in [0,1]$,

$$w(M_1) \geq y\frac{1}{2}\left(s_2(\pi^*) + \sum_{\ell=3}^{m} s_\ell(\pi^*)(\ell-1)\right) + (1-y)\left(|B_1| + \frac{1}{2}|B_2| + \frac{2}{3}|B_3|\right). \tag{2.6}$$

The permutation π produced by algorithm BP1 contains $b(\pi) \geq w(M_1) + \frac{1}{2}w'(M_2)$ bandpasses, as indicated at the end of Section 2.3.1. From Lemma 2.3, we have for any $x \in [0,1]$,

$$b(\pi) \geq w(M_1) + x\frac{1}{20}|B_2| + (1-x)\frac{1}{4}|B_4|. \tag{2.7}$$

Together with Eqs. (2.3) and (2.6), the above Eq. (2.7) becomes,

$$\begin{aligned} b(\pi) &\geq w(M_1) + x\frac{1}{20}|B_2| + (1-x)\frac{1}{4}|B_4| \\ &\geq y\frac{1}{2}\left(s_2(\pi^*) + \sum_{\ell=3}^{m} s_\ell(\pi^*)(\ell-1)\right) \\ &\quad +(1-y)\left(|B_1| + \frac{1}{2}|B_2| + \frac{2}{3}|B_3|\right) + x\frac{1}{20}|B_2| + (1-x)\frac{1}{4}|B_4| \end{aligned}$$

$$
\begin{aligned}
&= \frac{y}{2}\left(s_2(\pi^*) + \sum_{\ell=3}^{m} s_\ell(\pi^*)(\ell-1)\right) \\
&\quad +(1-y)|B_1| + \left(\frac{1-y}{2} + \frac{x}{20}\right)|B_2| + \frac{2(1-y)}{3}|B_3| + \frac{1-x}{4}|B_4| \\
&\geq \frac{5}{12}\left(s_2(\pi^*) + \sum_{\ell=3}^{m} s_\ell(\pi^*)(\ell-1)\right) + \frac{1}{18}|B_1| + \frac{1}{9}s_2(\pi^*), \qquad (2.8)
\end{aligned}
$$

where the last inequality is achieved by setting $x = \frac{5}{9}$ and $y = \frac{5}{6}$. Note that for all $\ell \geq 3$, $(\ell-1) \geq \frac{3}{2}\lfloor\frac{\ell}{2}\rfloor$. It then follows from Eqs. (2.8) and (2.1) that

$$
b(\pi) \geq \frac{19}{36}\left(s_2(\pi^*) + \frac{15}{19} \times \frac{3}{2}\sum_{\ell=3}^{m} s_\ell(\pi^*)\left\lfloor\frac{\ell}{2}\right\rfloor\right) \geq \frac{19}{36}b(\pi^*). \qquad (2.9)
$$

That is, the worst-case performance ratio of algorithm BP1 is at most $\frac{19}{36}$. □

2.4 $\frac{227}{426}$-approximation algorithm BP2

In this section, we introduce our second improved approximation algorithm BP2 for the *bandpass* problem.

2.4.1 Algorithm description

Again, the first step is to compute a maximum weight matching M_1 in graph G, every 1 involved in a bandpass of M_1 is changed to 0 and then construct a new graph G', which is the same as we described in the previous algorithm template in section 2.2.1.

In the second step of BP2, we compute a maximum weight 4-matching $\mathcal{C}$ in graph G', which is further decomposed in $O(n^{2.5})$ time into two 2-matchings denoted as $\mathcal{C}_1$ and $\mathcal{C}_2$ [37, 49]. Let $w'(\mathcal{C})$ denote the weight (the number of bandpasses) of $\mathcal{C}$ in the residual graph G'. Note that no bandpass of $\mathcal{C}$ shares a 1 with any bandpass of M_1. Using M_1 and $\mathcal{C}_1$ and $\mathcal{C}_2$, by Lemma 2.6, we can compute a matching M_2 from $\mathcal{C}$ of weight at least $\frac{1}{7.5}w'(\mathcal{C})$ such that $G[M_1 \cup M_2]$ is guaranteed acyclic.

In the third step, we use the $\frac{7}{9}$-approximation algorithm described in [78] to compute a Hamiltonian path $\mathcal{P}$ in G' whose weight is at least $\frac{7}{9}$ of the maximum weight of a Hamiltonian path. Then, using M_1 and $\mathcal{P}$, by Lemma 2.7, we can compute another matching M_2 from $\mathcal{P}$ of weight at least $\frac{1}{3}w'(\mathcal{P})$ such that $G[M_1 \cup M_2]$ is guaranteed acyclic.

In the last step, we choose the larger one between the two M_2's found in the last two steps, and arbitrarily stack the paths in $G[M_1 \cup M_2]$ to give a row permutation π. Note that the number of bandpasses extracted from π, $b(\pi)$, is greater than or equal to $w(M_1) + w'(M_2)$.

2.4.2 Performance analysis

In $O(n^{2.5})$ time, a 4-matching such as G'_s can be decomposed into two 2-matchings [37, 49], each of which is a collection of vertex-disjoint cycles or paths.

Lemma 2.5. *Let $\mathcal{C}$ be a 2-matching of graph G such that no edge of M_1 is also an edge of $\mathcal{C}$. Then, we can partition the edge set of $\mathcal{C}$ into four matchings X_0, X_1, X_2, X_3 such that $G[M_1 \cup X_j]$ is an acyclic 2-matching for all $j \in \{0, 1, 2, 3\}$. Moreover, the partitioning takes $O(n\alpha(n))$ time, where $\alpha(\cdot)$ is the inverse Ackerman function.*

Proof. Hassin and Rubinstein [50] have shown that we can compute two disjoint matchings X_0 and X_1 in $\mathcal{C}$ such that the following two conditions hold:

- Both $G[M_1 \cup X_0]$ and $G[M_1 \cup X_1]$ are acyclic 2-matchings of G.
- Each vertex of $\mathcal{C}$ is incident to at least one edge of $X_0 \cup X_1$.

For convenience, let Y be the set of edges in $\mathcal{C}$ but not in $X_0 \cup X_1$. By the second condition, Y is a matching. Consider the graph $H = (V, M_1 \cup Y)$. Obviously, H is a collection of vertex-disjoint paths and cycles, and each cycle of H contains at least two edges of Y. For each cycle C of H, we mark an arbitrary edge of C that also belongs to Y. Let X_3 be the set of marked edges, and $X_2 = Y \setminus X_3$. Then, both $G[M_1 \cup X_2]$ and $G[M_1 \cup X_3]$ are acyclic 2-matchings of G.

It is not hard to see that with the famous union-find data structure [97], the computation of X_0 and X_1 described in [50] can be done in $O(n\alpha(n))$ time. Once knowing X_0 and X_1, we can obtain X_2 and X_3 in $O(n)$ time. □

In general, Lemma 2.5 cannot be improved by partitioning the edge set of $\mathcal{C}$ into three matchings instead of four matchings. To see this, it suffices to consider a concrete example, where $\mathcal{C}$ is just a cycle of length 4 and M_1 consists of the two edges connecting nonadjacent vertices in $\mathcal{C}$.

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ denote the two 2-matchings constituting to the maximum weight 4-matching $\mathcal{C}$ of residual graph G'. Using Lemma 2.5 alone, $\mathcal{C}_1$ can be partitioned into four

matchings X_0, X_1, X_2, X_3 and $\mathcal{C}_2$ can be partitioned into four matchings Y_0, Y_1, Y_2, Y_3, such that $G[M_1 \cup Z_j]$ is an acyclic 2-matching for all $Z \in \{X, Y\}$ and $j \in \{0, 1, 2, 3\}$. The following lemma states a slightly better partition when we consider $\mathcal{C}_1$ and $\mathcal{C}_2$ simultaneously.

Lemma 2.6. *The weight of matching M_2 is $w'(M_2) \geq \frac{1}{15}|B_2|$.*

Proof. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ denote the two 2-matchings constituting to the maximum weight 4-matching $\mathcal{C}$ of residual graph G'. Based on the discussion in the last paragraph, we firstly use Lemma 2.5 to partition the edge set of $\mathcal{C}_1$ into four matchings X_0, X_1, X_2, X_3 and the edge set of $\mathcal{C}_2$ into four matchings Y_0, Y_1, Y_2, Y_3, such that $G[M_1 \cup Z_j]$ is an acyclic 2-matching for all $Z \in \{X, Y\}$ and $j \in \{0, 1, 2, 3\}$.

Note that by Lemma 2.5, $X_2 \cup X_3$ is a matching and that X_3 contains the marked edges, each of which, say $e = (u, v)$, is the lightest edge of the corresponding cycle, say C, formed in $G[M_1 \cup X_2 \cup X_3]$. C is an even cycle. If C contains at least 6 edges, then $w'(X_3 \cap C) = w'(e) \leq \frac{1}{2}w'(X_2 \cap C)$. The following process is to swap certain edges among X_0, X_1, X_2, X_3 and Y_0, Y_1, Y_2, Y_3 to guarantee the property

(P) that each of $G[M_1 \cup X_i]$ for $i = 0, 1$ and $G[M_1 \cup Y_j]$ for $j \in \{0, 1, 2, 3\}$ is an acyclic 2-matching, and that $X_2 \cup X_3$ is a matching and $G[M_1 \cup X_2 \cup X_3]$ contains no length-4 cycles.

Let $C = (u, v, x, y)$ be a length-4 cycle in $G[M_1 \cup X_2 \cup X_3]$, and assume that $X_2 \cup X_3 = \{(u, v), (x, y)\}$. Then, we call edges (u, v) and (x, y) a *problematic pair*. Our swapping process is to *resolve* such problematic pairs. We distinguish three cases.

In the first case, edges $(u, x) \notin \mathcal{C}_1$ and $(v, y) \notin \mathcal{C}_1$.

Assume the other edges of $\mathcal{C}_1$ incident at u, v, x, y are $(u, 1)$, $(v, 2)$, $(x, 3)$, $(y, 4)$, respectively. These four edges thus all belong to $G[M_1 \cup X_0]$ and $G[M_1 \cup X_1]$. If at least three of them belong to $G[M_1 \cup X_0]$, then in $G[M_1 \cup X_1]$ three vertices among u, v, x, y have degree 1 and thus they cannot be in the same connected component of $G[M_1 \cup X_1]$. We can move (exactly) one of edges (u, v) and (x, y) to X_1, while maintaining property (P).

We examine next where exactly two of the four edges belong to $G[M_1 \cup X_0]$. Assume without loss of generality that $(u, 1) \in G[M_1 \cup X_0]$. If $(y, 4) \in G[M_1 \cup X_0]$, then the connected component in $G[M_1 \cup X_1]$ containing u has only one edge (u, y), which belongs to M_1. Thus, if the other edge of $\mathcal{C}_1$ incident at vertex 1 belongs to X_1, we can move edge $(u, 1)$ from X_0 to $X_2 \cup X_3$, and move edge (u, v) from $X_2 \cup X_3$ to X_0; if the other edge of $\mathcal{C}_1$ incident at vertex 1 does not belong to X_1 (and thus it must be in $X_2 \cup X_3$),

we can move edge $(u, 1)$ from X_0 to X_1, and move edge (u, v) from $X_2 \cup X_3$ to X_0. Either way, we maintain property (P) while resolving a problematic pair of $X_2 \cup X_3$.

If $(v, 2) \in G[M_1 \cup X_0]$, then vertices u and v have degree 1 in $G[M_1 \cup X_1]$. Thus, if the other edge of $\mathcal{C}_1$ incident at vertex 1 does not belong to X_1, then vertex 1 has degree 1 in $G[M_1 \cup X_1]$ as well. We conclude that vertices $u, v, 1$ cannot reside in the same connected component of $G[M_1 \cup X_1]$. When u and v are not connected, we can move edge (u, v) from $X_2 \cup X_3$ to X_1; when u and 1 are not connected, we can move edge $(u, 1)$ from X_0 to X_1, and move edge (x, y) from $X_2 \cup X_3$ to X_0. Again, either way, we maintain property (P) while resolving a problematic pair of $X_2 \cup X_3$. Symmetric scenarios can be argued in the same way for vertices $2, 3, 4$. In the remaining scenario, the other edges of $\mathcal{C}_1$ incident at vertices $1, 2, 3, 4$ all belong to $X_0 \cup X_1$. We then move edges $(u, 1), (v, 2), (x, 3), (y, 4)$ from $X_0 \cup X_1$ to $X_2 \cup X_3$, and move edges (u, v) ((x, y), respectively) from $X_2 \cup X_2$ to X_0 (X_3, respectively). Note that none of these four edges would form with any other edge into a problematic pair.

Lastly, if $(x, 3) \in G[M_1 \cup X_0]$, then vertices u and x have degree 1 in $G[M_1 \cup X_1]$. Thus, if the other edge of $\mathcal{C}_1$ incident at vertex 1 belongs to X_1, then vertex 1 has degree 1 in $G[M_1 \cup X_2 \cup X_3]$. We can move edge $(u, 1)$ from X_0 to $X_2 \cup X_3$, and move edge (u, v) from $X_2 \cup X_3$ to X_0. If the other edge of $\mathcal{C}_1$ incident at vertex 1 does not belong to X_1, then vertex 1 has degree 1 in $G[M_1 \cup X_1]$ as well. We conclude that vertices $u, x, 1$ cannot reside in the same connected component of $G[M_1 \cup X_1]$. When u and 1 are not connected, we can move edge $(u, 1)$ from X_0 to X_1, and move edge (u, v) from $X_2 \cup X_3$ to X_0. Symmetric scenarios can be argued in the same way for vertices $2, 3, 4$. In the remaining scenario, none of the other edges of $\mathcal{C}_1$ incident at vertices $1, 2, 3, 4$ belongs to $X_0 \cup X_1$, and that vertices u and 1 (v and 2, x and 3, y and 4, respectively) are connected in $G[M_1 \cup X_1]$ ($G[M_1 \cup X_0]$, $G[M_1 \cup X_1]$, $G[M_1 \cup X_0]$, respectively). It follows that we may move edge $(u, 1)$ from X_0 to X_1, move edge $(y, 4)$ from X_1 to X_0, and move edge (u, v) from $X_2 \cup X_3$ to X_0, to resolve the problematic pair.

In the second case, edges $(u, x) \notin \mathcal{C}_1$ but $(v, y) \in \mathcal{C}_1$.

Assume the other edges of $\mathcal{C}_1$ incident at u, x are $(u, 1)$, $(x, 3)$, respectively. These two edges and edge (v, y) all belong to $G[M_1 \cup X_0]$ and $G[M_1 \cup X_1]$. Without loss of generality, assume $(v, y) \in X_1$; it follows that vertices v and y have degree 1 in $G[M_1 \cup X_0]$. If one of edges $(u, 1)$ and $(x, 3)$ does not belong to $G[M_1 \cup X_0]$, say $(u, 1)$, then we can move (u, v) from $X_2 \cup X_3$ to X_0, while maintaining property (P).

If both edges $(u, 1)$ and $(x, 3)$ belong to $G[M_1 \cup X_0]$, then vertices u and x have degree 1 in $G[M_1 \cup X_1]$. When the other edge of $\mathcal{C}_1$ incident at vertex 1 does not belong to X_1 (but $X_2 \cup X_3$), we can move edge $(u, 1)$ from X_0 to X_1, and move edge (u, v) from

$X_2 \cup X_3$ to X_0; the symmetric scenario can be argued in the same way for vertex 3; When the other edge of $\mathcal{C}_1$ incident at vertex 1 and the other edge of $\mathcal{C}_1$ incident at vertex 3 both belong to X_1, we can move edges $(u, 1)$ and $(v, 3)$ from X_0 to $X_2 \cup X_3$, move edge (v, y) from X_1 to $X_2 \cup X_3$, move edge (u, v) from $X_2 \cup X_3$ to X_0, and move edge (x, y) from $X_2 \cup X_3$ to X_1. Note that none of these three edges $(u, 1)$, $(v, 3)$ and (v, y) would form with any other edge into a problematic pair.

In the last case, edges $(u, x) \in \mathcal{C}_1$ and $(v, y) \in \mathcal{C}_1$.

Assume without loss of generality that $(u, x) \in X_0$ and $(v, y) \in X_1$. Since $\mathcal{C}_2$ do not share any edge with $\mathcal{C}_1$, we consider the degrees of vertices u, v, x, y in $G[M_1 \cup Y_i]$ for $i = 0, 1, 2, 3$. If in one of these four acyclic 2-matchings, say $G[M_1 \cup Y_0]$, at least three of the four vertices have degree 1, say u, v, x, then we can move edge (u, v) from $\mathcal{C}_1$ to Y_0, and thus the problematic pair of $X_2 \cup X_3$ is resolved. In the other cases, in each $G[M_1 \cup Y_i]$ for $i = 0, 1, 2, 3$, exactly two of the four vertices have degree 1.

Let the two edges of $\mathcal{C}_2$ incident at u (v, x, y, respectively) be $(u, 1)$ and $(u, 1')$ ($(v, 2)$ and $(v, 2')$, $(x, 3)$ and $(x, 3')$, $(y, 4)$ and $(y, 4')$, respectively).

If $(u, 1), (y, 4) \in Y_0$, then u and y both have degree 1 in one of $G[M_1 \cup Y_i]$ for $i = 1, 2, 3$, say in $G[M_1 \cup Y_3]$. It follows that if the other edge of $\mathcal{C}_2$ incident at vertex 1 does not belong to Y_3, then we can move edge $(u, 1)$ from Y_0 to Y_3, and move edge (u, v) from $\mathcal{C}_1$ to Y_0 to resolve the problematic pair of $X_2 \cup X_3$; or if the other edge of $\mathcal{C}_2$ incident at vertex 4 does not belong to Y_3, then we can move edge $(y, 4)$ from Y_0 to Y_3, and move edge (x, y) from $\mathcal{C}_1$ to Y_0 to resolve the problematic pair of $X_2 \cup X_3$. In the remaining scenario, the other edge of $\mathcal{C}_2$ incident at vertex 1 (vertex 4, respectively) belongs to Y_3. Note that in either $G[M_1 \cup Y_1]$ or $G[M_1 \cup Y_2]$, vertex u has degree 1, and we assume without loss of generality that vertex u has degree 1 in $G[M_1 \cup Y_1]$. Note also that vertex 1 has degree 1 in $G[M_1 \cup Y_1]$. If edge $(y, 4') \notin Y_1$, then vertex y has degree 1 as well, and thus we can move edge $(u, 1)$ from Y_0 to Y_1, and move edge (u, v) from $\mathcal{C}_1$ to Y_0 to resolve the problematic pair of $X_2 \cup X_3$; if edge $(y, 4') \in Y_1$ but the other edge of $\mathcal{C}_2$ incident at vertex $4'$ does not belong to Y_3, then we can move edge $(y, 4')$ from Y_1 to Y_3, move edge $(u, 1)$ from Y_0 to Y_1, and move edge (u, v) from $\mathcal{C}_1$ to Y_0 to resolve the problematic pair of $X_2 \cup X_3$. Therefore, we only need to argue the scenario where the other edge of $\mathcal{C}_2$ incident at vertex $4'$ belongs to Y_3. Symmetrically considering Y_2, we may assume without loss of generality that the other edge of $\mathcal{C}_2$ incident at vertex $1'$ belongs to Y_3. Consequently, vertices $u, 1, 1'$ all have degree 1 in $G[M_1 \cup Y_1]$, and thus u and at least one of 1 and $1'$ are not connected. If u and 1 are not connected, we can move edge $(u, 1)$ from Y_0 to Y_1, and move edge (u, v) from $\mathcal{C}_1$ to Y_0 to resolve the problematic pair of $X_2 \cup X_3$; if u and $1'$ are not connected, we can move edge $(u, 1')$

from Y_2 to Y_1, move edge $(u, 1)$ from Y_0 to Y_2, and move edge (u, v) from $\mathcal{C}_1$ to Y_0 to resolve the problematic pair of $X_2 \cup X_3$.

If $(u, 1), (v, 2) \in Y_0$, then u and v both have degree 1 in one of $G[M_1 \cup Y_i]$ for $i = 1, 2, 3$, say in $G[M_1 \cup Y_3]$. The following discussion is very similar to the above paragraph, though slightly simpler. Firstly, if x and y are not connected in $G[M_1 \cup Y_0]$ (u and v are not connected in $G[M_1 \cup Y_3]$, respectively), then we can move edge (x, y) ((u, v), respectively) from $\mathcal{C}_1$ to Y_0 (Y_3, respectively) to directly resolve the problematic pair of $X_2 \cup X_3$. Secondly, if the other edge of $\mathcal{C}_2$ incident at vertex 1 does not belong to Y_3, then we can move edge $(u, 1)$ from Y_0 to Y_3, and move edge (x, y) from $\mathcal{C}_1$ to Y_0 to resolve the problematic pair of $X_2 \cup X_3$; or if the other edge of $\mathcal{C}_2$ incident at vertex 2 does not belong to Y_3, then we can move edge $(v, 2)$ from Y_0 to Y_3, and move edge (x, y) from $\mathcal{C}_1$ to Y_0 to resolve the problematic pair of $X_2 \cup X_3$. Symmetrically and without loss of generality that $(x, 3), (y, 4) \in Y_3$, if either of the other edges of $\mathcal{C}_2$ incident at vertices 3 and 4 does not belong to Y_3, the problematic pair can be resolved. In the remaining scenario, we assume that vertices u and x have degree 1 in $G[M_1 \cup Y_1]$ (and $(v, 2'), (y, 4') \in Y_1$). Note that vertices $1, 2, 3, 4$ all have degree 1 in $G[M_1 \cup Y_1]$ too. If u and x are not connected in $G[M_1 \cup Y_1]$, then we can swap edges of $X_0 \cup X_1$ and of $X_2 \cup X_3$, and move edge (u, x) from $X_2 \cup X_3$ to Y_1, to resolve the problematic pair of $X_2 \cup X_3$. Otherwise, u and 1 should not be connected in $G[M_1 \cup Y_1]$, and we can move edge $(u, 1)$ from Y_0 to Y_1, and move edge (x, y) from $X_2 \cup X_3$ to Y_0, to resolve the problematic pair of $X_2 \cup X_3$.

All the other pairs of edges occurring in $\mathcal{C}_2 \cap Y_0$ can be analogously discussed as in either of the above two paragraphs. Repeatedly applying the above process to resolve the problematic pairs of $X_2 \cup X_3$, if any, we achieve the Property (P) that each of $G[M_1 \cup X_i]$ for $i = 0, 1$ and $G[M_1 \cup Y_j]$ for $j \in \{0, 1, 2, 3\}$ is an acyclic 2-matching, and that $X_2 \cup X_3$ is a matching and $G[M_1 \cup X_2 \cup X_3]$ contains no length-4 cycles. Subsequently, we let X_3 denote the set of marked edges, guaranteeing that $w'(X_3) \leq \frac{1}{2} w'(X_2)$.

It follows that at least one of $X_0, X_1, X_2, Y_0, Y_1, Y_2, Y_3$ has its weight greater than or equal to

$$\frac{1}{7.5} \big(w'(\mathcal{C}_1) + w'(\mathcal{C}_2)\big) \geq \frac{1}{7.5} \times \frac{1}{2} |B_2| = \frac{1}{15} |B_2|,$$

where the last inequality follows from Lemma 2.2 and the fact that $w'(\mathcal{C}) \geq w'(G'_s)$. □

The next lemma says that Lemma 2.5 can be improved if the input 2-matching is acyclic.

Lemma 2.7. *Let $\mathcal{P}$ be an acyclic 2-matching of G such that no edge of M_1 is also an edge of $\mathcal{P}$. Then, we can partition the edge set of $\mathcal{P}$ into three matchings Y_0, Y_1, Y_2 such that $G[M_1 \cup Y_j]$ is an acyclic 2-matching for all $j \in \{0, 1, 2\}$. Moreover, the partitioning takes $O(n\alpha(n))$ time.*

Proof. Note that $\mathcal{P}$ is a collection of vertex-disjoint paths. We claim that if $\mathcal{P}$ has two or more connected components, then we can connect the connected components of $\mathcal{P}$ into a single path by adding edges not in M_1 to $\mathcal{P}$. To see this claim, suppose that $\mathcal{P}$ has two or more connected components. Obviously, we can connect the connected components of $\mathcal{P}$ into a single path by adding edges to $\mathcal{P}$. Unfortunately, some edges of M_1 may have been added to $\mathcal{P}$. To remove edges of M_1 from $\mathcal{P}$, we start at one endpoint of $\mathcal{P}$ and process the edges of $\mathcal{P}$ in order as follows:

- Let s and t be the current endpoints of $\mathcal{P}$, and (u, v) be the current edge we want to process. Without loss of generality, we may assume that the removal of (u, v) from $\mathcal{P}$ yields a path $\mathcal{P}_u$ from s to u and another path $\mathcal{P}_v$ from v to t, and further assume that the edges of $\mathcal{P}_u$ have been processed. Note that at most one of $s = u$ and $v = t$ is possible because $n \geq 3$. If $(u, v) \notin M_1$, then we proceed to process the other edge incident to v than (u, v). Otherwise, $(v, s) \notin M_1$ or $(u, t) \notin M_1$ because M_1 is a matching and at most one of $s = u$ and $v = t$ is possible. If $(v, s) \notin M_1$, then we modify $\mathcal{P}$ by deleting edge (u, v) and adding edge (v, s) and proceed to process the other edge incident to v than (v, s). On the other hand, if $(u, t) \notin M_1$, then we modify $\mathcal{P}$ by deleting edge (u, v) and adding edge (u, t) and proceed to process the other edge incident to t than (u, t).

By the above claim, we may assume that $\mathcal{P}$ is a single path $\mathcal{P} = (v_1, v_2, \ldots, v_{\ell+1})$, and denote $e_j = (v_j, v_{j+1})$ for $j = 1, 2, \ldots, \ell$.

We next detail how to partition the edge set of $\mathcal{P}$ into three required matchings Y_0, Y_1, and Y_2. Initially, we set $Y_0 = \{e_1\}$, $Y_1 = \{e_2\}$, and $Y_2 = \{e_3\}$. Then, for $j = 4, 5, \ldots, \ell$ (in this order), we try to find a $k \in \{0, 1, 2\}$ such that $Y_k \cup \{e_j\}$ is a matching and $G[M_1 \cup Y_k \cup \{e_j\}]$ is an acyclic 2-matching of G. To explain how to find k, fix an integer $j \in \{4, 5, \ldots, \ell\}$. Let b be the integer in $\{0, 1, 2\}$ with $e_{j-1} \in Y_b$, and b' and b'' be the two integers in $\{0, 1, 2\} \setminus \{b\}$. If $G[M_1 \cup Y_{b'}]$ (respectively, $G[M_1 \cup Y_{b''}]$) contains no path between v_j and v_{j+1}, then we can set $k = b'$ (respectively, $k = b''$) and we are done. So, we may also assume that $G[M_1 \cup Y_{b'}]$ contains a path P' between v_j and v_{j+1} and $G[M_1 \cup Y_{b''}]$ contains a path P'' between v_j and v_{j+1}. See FIGURE 2.2.

Let $v_{i'}$ (respectively, $v_{i''}$) be the neighbor of v_j in P' (respectively, P''), and $v_{h'}$ (respectively, $v_{h''}$) be the neighbor of v_{j+1} in P' (respectively, P''). Then, none of edges (v_{j-1}, v_j), (v_j, v_{j+1}), and (v_{j+1}, v_{j+2}) can appear in P' (respectively, P''), because $(v_{j-1}, v_j) \in Y_b$ and neither (v_j, v_{j+1}) nor (v_{j+1}, v_{j+2}) has been processed. So, all of $(v_j, v_{i'})$, $(v_{j+1}, v_{h'})$, $(v_j, v_{i''})$, and $(v_{j+1}, v_{h''})$ belong to M_1. Thus, $i' = i''$ and $h' = h''$ because M_1 is a matching. Consequently, one edge incident to $v_{i'}$ (respectively, $v_{h'}$) in $\mathcal{P}$ belongs to $Y_{b'}$ and the other belongs to $Y_{b''}$. Hence, $i' < j - 1$ and $h' < j - 1$.

Since $e_{j-1} \in Y_b$, either $e_{j-2} \in Y_{b'}$ or $e_{j-2} \in Y_{b''}$. We assume that $e_{j-2} \in Y_{b'}$; the case where $e_{j-2} \in Y_{b''}$ is similar. Since P'' is a path between v_j and v_{j+1} in $G[M_1 \cup Y_{b''}]$, $G[M_1 \cup Y_{b''}]$ contains no path between v_j and v_{j-1}. Thus, $G[M_1 \cup Y_{b''} \cup \{e_{j-1}\}]$ is an acyclic 2-matching of G. Hence, we move e_{j-1} from Y_b to $Y_{b''}$. A crucial point is that the degree of $v_{i'}$ in $G[M_1 \cup Y_b]$ is 1. This is true, because $v_{i'}$ appears in both P' and P'' and in turn cannot be incident to an edge in Y_b. By this crucial point and the fact that $v_{i'}$ and v_j belong to the same connected component in $G[M_1 \cup Y_b \cup \{e_j\}]$, we know that $G[M_1 \cup Y_b \cup \{e_j\}]$ is an acyclic 2-matching of G. Therefore, we can set $k = b$.

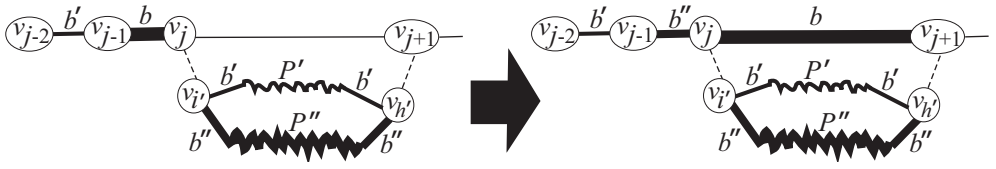

FIGURE 2.2: An illustration of moving (v_{j-1}, v_j) from Y_b to $Y_{b''}$ and adding (v_j, v_{j+1}) to Y_b, where (1) the dashed lines indicate edges in M_1, (2) the thin solid lines indicate edges of $\mathcal{P}$ that have not been processed, (3) the lines labeled with b (respectively, b', or b'') indicate edges in Y_b (respectively, $Y_{b'}$, or $Y_{b''}$), and (4) the two curves may contain edges of M_1.

Obviously, with the famous union-find data structure [97], the above partitioning of the edge set $\mathcal{P}$ into Y_0, Y_1, Y_2 can be done in $O(n\alpha(n))$ time. □

In general, Lemma 2.7 cannot be improved by partitioning the edge set of $\mathcal{P}$ into two matchings instead of three matchings. To see this, it suffices to consider a concrete example, where $\mathcal{P}$ is just a path with edges (v_1, v_2), (v_2, v_3), (v_3, v_4) and M_1 consists of edges (v_1, v_3) and (v_2, v_4).

Lemma 2.8. *The weight of matching M_2 is $w'(M_2) \geq \frac{7}{27}|B_4|$.*

Proof. Note that graph G' contains all bandpasses of B_4, which is an acyclic 2-matching. By the 7/9-approximation algorithm for the Max-TSP [78], we can compute a Hamiltonian path $\mathcal{P}$ in G' of weight at least $\frac{7}{9}$ of the optimum, and thus of weight at least $\frac{7}{9}|B_4|$. The above Lemma 2.7 guarantees that

$$w'(M_2) \geq \frac{1}{3}w'(\mathcal{P}) \geq \frac{7}{27}|B_4|.$$

□

Theorem 2.9. *Algorithm* BP2 *is an $O(n^4)$-time $\frac{227}{426}$-approximation for the bandpass problem.*

Proof. The running time of algorithm BP2 is dominated by the computing for those maximum weight b-matchings, for $b = 1, 2, 4$, which can be done in $O(n^4)$ time. Since

M_1 is the maximum weight matching in graph G, from Eq. (2.1) we have

$$w(M_1) \geq \frac{1}{2}p(\pi^*) \geq \frac{1}{2}\left(s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell-1)\right). \tag{2.10}$$

Combining Eqs. (2.4) and (2.10), we have for any real number $y \in [0,1]$,

$$w(M_1) \geq y\frac{1}{2}\left(s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell-1)\right) + (1-y)\left(|B_1| + \frac{1}{2}|B_2| + \frac{2}{3}|B_3|\right). \tag{2.11}$$

The permutation π produced by algorithm BP2 contains $b(\pi) \geq w(M_1) + w'(M_2)$ bandpasses, as indicated at the end of Section 2.4.1. From Lemmas 2.6 and 2.8, we have for any real number $x \in [0,1]$,

$$b(\pi) \geq w(M_1) + x\frac{1}{15}|B_2| + (1-x)\frac{7}{27}|B_4|. \tag{2.12}$$

Together with Eqs. (2.3) and (2.11), the above Eq. (2.12) becomes,

$$\begin{aligned}
b(\pi) &\geq w(M_1) + x\frac{1}{15}|B_2| + (1-x)\frac{7}{27}|B_4| \\
&\geq y\frac{1}{2}\left(s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell-1)\right) \\
&\quad +(1-y)\left(|B_1| + \frac{1}{2}|B_2| + \frac{2}{3}|B_3|\right) + x\frac{1}{15}|B_2| + (1-x)\frac{7}{27}|B_4| \\
&= \frac{y}{2}\left(s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell-1)\right) \\
&\quad +(1-y)|B_1| + \left(\frac{1-y}{2} + \frac{x}{15}\right)|B_2| + \frac{2(1-y)}{3}|B_3| + \frac{7(1-x)}{27}|B_4| \\
&\geq \frac{57}{142}\left(s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell-1)\right) + \frac{14}{213}|B_1| + \frac{28}{213}s_2(\pi^*),
\end{aligned} \tag{2.13}$$

where the last inequality is achieved by setting $x = \frac{35}{71}$ and $y = \frac{57}{71}$. Note that for all $\ell \geq 3$, $(\ell-1) \geq \frac{3}{2}\lfloor\frac{\ell}{2}\rfloor$. It then follows from Eqs. (2.13) and (2.1) that

$$b(\pi) \geq \frac{227}{426}\left(s_2(\pi^*) + \frac{171}{227} \times \frac{3}{2}\sum_{\ell=3}^{n} s_\ell(\pi^*)\left\lfloor\frac{\ell}{2}\right\rfloor\right) \geq \frac{227}{426}b(\pi^*). \tag{2.14}$$

That is, the worst-case performance ratio of algorithm BP2 is at most $\frac{227}{426}$. □

2.5 $\frac{70-\sqrt{2}}{128}$-approximation algorithm BP3

In this section, we will introduce our third approximation algorithm for the *bandpass* problem, which is also the currently best approximation algorithm for the the *bandpass* problem.

2.5.1 Algorithm description

BP3 is very similar to the algorithm BP2. The main difference is that BP3 presents another scheme to partition a 4-matching into a number of candidate submatchings, each of which can be used to extend the first maximum weight matching. Again, the first step of BP3 is the same as the first step of the algorithm template described in the Section 2.2.1.

In the second step of BP3, we compute a maximum weight 4-matching $\mathcal{C}$ in graph G', which is then decomposed in $O(n^{2.5})$ time into two 2-matchings denoted as $\mathcal{C}_1$ and $\mathcal{C}_2$ [37, 49]. Let $w'(\mathcal{C})$ denote the total residual weight of all the edges in $\mathcal{C}$. Using M_1 and the two 2-matchings $\mathcal{C}_1$ and $\mathcal{C}_2$, by Lemma 2.16, we can compute a matching M_2 out of $\mathcal{C}$, of weight at least $\frac{6-\sqrt{2}}{32}w'(\mathcal{C}) > 0.1433w'(\mathcal{C})$, to extend M_1 into an acyclic 2-matching.

We then arbitrarily stack the paths in $G[M_1 \cup M_2]$ to give a solution row permutation π. Note that the number of bandpasses extracted from π, $b(\pi)$, is greater than or equal to $w(M_1) + w'(M_2)$.

2.5.2 Performance analysis

The following lemma is a restatement of Lemma 2.5.

Lemma 2.10. *Let $\mathcal{C}$ be a 2-matching of graph G such that $M_1 \cap \mathcal{C} = \emptyset$. Then, we can partition the edge set of $\mathcal{C}$ into four matchings X_0, X_1, X_2, X_3 such that*

i) *$G[M_1 \cup X_j]$ is an acyclic 2-matching for all $j \in \{0, 1, 2, 3\}$;*

ii) *each vertex of $\mathcal{C}$ is incident to at least one edge of $X_0 \cup X_1$ (and thus $X_2 \cup X_3$ is a matching);*

iii) *X_3 contains exactly the lightest edge of $X_2 \cup X_3$ in every cycle of $G[M_1 \cup X_2 \cup X_3]$.*

Moreover, the partitioning takes $O(n\alpha(n))$ time, where $\alpha(\cdot)$ is the inverse Ackerman function.

The correctness of Lemma 2.10 follows from the proof of the Lemma 2.5.

Lemma 2.11. *Let X_3' denote the subset of X_3 of which each edge comes from a length-4 cycle of $G[M_1 \cup X_2 \cup X_3]$, and let $X_3'' = X_3 - X_3'$. Then each edge of X_3'' comes from a cycle of $G[M_1 \cup X_2 \cup X_3]$ of length at least 6.*

Proof. Since $X_2 \cup X_3$ is a matching, every cycle of $G[M_1 \cup X_2 \cup X_3]$ is an even cycle. The lemma follows clearly, since there are no multiple edges in $G[M_1 \cup X_2 \cup X_3]$. □

Lemma 2.12. *Let X_2' denote the subset of X_2 of which each edge comes from a length-4 cycle of $G[M_1 \cup X_2 \cup X_3]$, and let $X_2'' = X_2 - X_2'$. Then $w'(X_3') \leq w'(X_2')$ and $w'(X_3'') \leq \frac{1}{2}w'(X_2'')$.*

Proof. The lemma follows clearly, since every edge of X_3 is the lightest edge of $X_2 \cup X_3$ in some cycle of $G[M_1 \cup X_2 \cup X_3]$. □

The following Lemmas 2.13 and 2.15 state slightly better partition results than Lemma 2.10.

Lemma 2.13. *If $\mathcal{C}_1$ contains no length-4 cycle, then we can partition $\mathcal{C}_1$ into four matchings X_0, X_1, X_2 and X_3 as described in* Lemma 2.10*, and such that $X_3' = \emptyset$ (i.e., $G[M_1 \cup X_2 \cup X_3]$ contains no length-4 cycle). Moreover, the partitioning takes $O(n\alpha(n))$ time.*

Proof. We firstly use Lemma 2.10 to partition the edge set of $\mathcal{C}_1$ into four matchings X_0, X_1, X_2, X_3. The following process is to swap certain edges among X_0, X_1, X_2, X_3 to guarantee all three properties stated in Lemma 2.10, plus a novel property that $G[M_1 \cup X_2 \cup X_3]$ contains no length-4 cycles.

Let $C = (u, v, x, y)$ be a length-4 cycle in $G[M_1 \cup X_2 \cup X_3]$, and assume that edges $(u, v), (x, y) \in X_2 \cup X_3$ and edges $(u, y), (v, x) \in M_1$. We call edges (u, v) and (x, y) a *problematic pair*. Our swapping process is to *resolve* such problematic pairs. We distinguish two cases.

In the first case, edges $(u, x) \notin \mathcal{C}_1$ and $(v, y) \notin \mathcal{C}_1$.

Assume the other edges of $\mathcal{C}_1$ incident at u, v, x, y are $(u, 1)$, $(v, 2)$, $(x, 3)$, $(y, 4)$, respectively. These four edges thus all belong to $G[M_1 \cup X_0]$ and $G[M_1 \cup X_1]$. If at least three of them belong to $G[M_1 \cup X_0]$, then in $G[M_1 \cup X_1]$ three vertices among u, v, x, y have degree 1 and thus they cannot be in the same connected component of $G[M_1 \cup X_1]$. We can move (exactly) one of edges (u, v) and (x, y) from $X_2 \cup X_3$ to X_1, while resolving this problematic pair.

We examine next where exactly two of the four edges belong to $G[M_1 \cup X_0]$. Assume without loss of generality that $(u,1) \in G[M_1 \cup X_0]$. If $(y,4) \in G[M_1 \cup X_0]$, then the connected component in $G[M_1 \cup X_1]$ containing u has only one edge (u,y), which belongs to M_1. Thus, if the other edge of $\mathcal{C}_1$ incident at vertex 1 belongs to X_1, we can move edge $(u,1)$ from X_0 to $X_2 \cup X_3$, and move edge (u,v) from $X_2 \cup X_3$ to X_0; if the other edge of $\mathcal{C}_1$ incident at vertex 1 does not belong to X_1 (and thus it must be in $X_2 \cup X_3$), we can move edge $(u,1)$ from X_0 to X_1, and move edge (u,v) from $X_2 \cup X_3$ to X_0. Either way, we maintain all three properties stated in Lemma 2.10 while resolving this problematic pair.

If $(v,2) \in G[M_1 \cup X_0]$, then both vertices u and v have degree 1 in $G[M_1 \cup X_1]$. Thus, if the other edge of $\mathcal{C}_1$ incident at vertex 1 does not belong to X_1, then vertex 1 has degree 1 in $G[M_1 \cup X_1]$ as well. We conclude that vertices $u, v, 1$ cannot reside in the same connected component of $G[M_1 \cup X_1]$. When u and v are not connected, we can move edge (u,v) from $X_2 \cup X_3$ to X_1; when u and 1 are not connected, we can move edge $(u,1)$ from X_0 to X_1, and move edge (x,y) from $X_2 \cup X_3$ to X_0. Again, either way, we maintain all three properties stated in Lemma 2.10 while resolving this problematic pair. Symmetric scenarios can be argued in the same way for vertices $2, 3, 4$. In the remaining scenario, the other edges of $\mathcal{C}_1$ incident at vertices $1, 2, 3, 4$ all belong to $X_0 \cup X_1$. We then move edges $(u,1), (v,2), (x,3), (y,4)$ from $X_0 \cup X_1$ to $X_2 \cup X_3$, and move edges (u,v) ((x,y), respectively) from $X_2 \cup X_3$ to X_0 (X_1, respectively). Note that none of these four edges would form with any other edge into a problematic pair.

Lastly, if $(x,3) \in G[M_1 \cup X_0]$, then vertices u and x have degree 1 in $G[M_1 \cup X_1]$. Thus, if the other edge of $\mathcal{C}_1$ incident at vertex 1 belongs to X_1, then vertex 1 has degree 1 in $G[M_1 \cup X_2 \cup X_3]$. We can move edge $(u,1)$ from X_0 to $X_2 \cup X_3$, and move edge (u,v) from $X_2 \cup X_3$ to X_0. If the other edge of $\mathcal{C}_1$ incident at vertex 1 does not belong to X_1, then vertex 1 has degree 1 in $G[M_1 \cup X_1]$ as well. We conclude that vertices $u, x, 1$ cannot reside in the same connected component of $G[M_1 \cup X_1]$. When u and 1 are not connected, we can move edge $(u,1)$ from X_0 to X_1, and move edge (u,v) from $X_2 \cup X_3$ to X_0. Symmetric scenarios can be argued in the same way for vertices $2, 3, 4$. In the remaining scenario, none of the other edges of $\mathcal{C}_1$ incident at vertices $1, 2, 3, 4$ belongs to $X_0 \cup X_1$, and that vertices u and 1 (v and 2, x and 3, y and 4, respectively) are connected in $G[M_1 \cup X_1]$ ($G[M_1 \cup X_0]$, $G[M_1 \cup X_1]$, $G[M_1 \cup X_0]$, respectively). It follows that we may move edge $(u,1)$ from X_0 to X_1, move edge $(y,4)$ from X_1 to X_0, and move edge (u,v) from $X_2 \cup X_3$ to X_0, to resolve the problematic pair.

In the second case, edges $(u,x) \notin \mathcal{C}_1$ but $(v,y) \in \mathcal{C}_1$.

Assume the other edges of $\mathcal{C}_1$ incident at u, x are $(u,1)$, $(x,3)$, respectively. These two edges and edge (v,y) all belong to $G[M_1 \cup X_0]$ and $G[M_1 \cup X_1]$. Without loss of generality,

assume $(v, y) \in X_1$; it follows that both vertices v and y have degree 1 in $G[M_1 \cup X_0]$. If edge $(u, 1)$ (edge $(x, 3)$, respectively) does not belong to $G[M_1 \cup X_0]$, then we can move (u, v) ((x, y), respectively) from $X_2 \cup X_3$ to X_0 to resolve the problematic pair.

If both edges $(u, 1)$ and $(x, 3)$ belong to $G[M_1 \cup X_0]$, then both vertices u and x have degree 1 in $G[M_1 \cup X_1]$. When the other edge of $\mathcal{C}_1$ incident at vertex 1 does not belong to X_1 (but $X_2 \cup X_3$), we can move edge $(u, 1)$ from X_0 to X_1, and move edge (u, v) from $X_2 \cup X_3$ to X_0; the symmetric scenario can be argued in the same way for vertex 3; When the other edge of $\mathcal{C}_1$ incident at vertex 1 and the other edge of $\mathcal{C}_1$ incident at vertex 3 both belong to X_1, we can move edges $(u, 1)$ and $(v, 3)$ from X_0 to $X_2 \cup X_3$, move edge (v, y) from X_1 to $X_2 \cup X_3$, move edge (u, v) from $X_2 \cup X_3$ to X_0, and move edge (x, y) from $X_2 \cup X_3$ to X_1. These movings maintain all three properties stated in Lemma 2.10 while resolving the problematic pair. Note that none of these three edges $(u, 1)$, $(v, 3)$ and (v, y) would form with any other edge into a problematic pair.

Note that it is impossible to have both edges $(u, x) \in \mathcal{C}_1$ and $(v, y) \in \mathcal{C}_1$, as they imply a length-4 cycle (u, v, x, y) in $\mathcal{C}_1$ contains. Repeatedly applying the above process to resolve the problematic pairs of $X_2 \cup X_3$, if any, we achieve the desired partitioning stated in Lemma 2.10, with the extra property that $G[M_1 \cup X_2 \cup X_3]$ contains no length-4 cycles. Therefore, $X_3' = \emptyset$. □

Lemma 2.14. *Let C be a length-4 cycle in a 2-matching $\mathcal{C}$ such that $M_1 \cap \mathcal{C} = \emptyset$. Then we can partition $\mathcal{C}$ into four matchings X_0, X_1, X_2 and X_3 as described in* Lemma 2.10*, and such that the lightest edge of C is assigned to $X_2 \cup X_3$.*

Proof. The lemma follows trivially from the observation that exactly two non-adjacent edges of C have to be assigned to $X_2 \cup X_3$, and either way is feasible. □

Lemma 2.15. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two disjoint 2-matchings of graph G such that $M_1 \cap (\mathcal{C}_1 \cup \mathcal{C}_2) = \emptyset$. Then, we can partition $\mathcal{C}_1$ into four matchings X_0, X_1, X_2, X_3 and partition $\mathcal{C}_2$ into four matchings Y_0, Y_1, Y_2, Y_3 such that the partition of $\mathcal{C}_1$ satisfies all the desired properties described in* Lemma 2.10 *plus $X_3' = \emptyset$ (i.e., $G[M_1 \cup X_2 \cup X_3]$ contains no length-4 cycle), the partition of $\mathcal{C}_2$ satisfies that $G[M_1 \cup Y_i]$ is an acyclic 2-matching for all $i \in \{0, 1, 2, 3\}$. Moreover, the partitioning takes $O(n\alpha(n))$ time.*

Proof. We firstly partition $\mathcal{C}_1$ ($\mathcal{C}_2$, respectively) into four matchings X_0, X_1, X_2, X_3 (Y_0, Y_1, Y_2, Y_3, respectively) to satisfies all the desired properties described in Lemma 2.10. If there is no length-4 cycle in $\mathcal{C}_1$, then Lemma 2.13 implies the current lemma. Otherwise, for each problematic pair of edges in a length-4 cycle of $G[M_1 \cup X_2 \cup X_3]$, we can resolve it as in the proof of Lemma 2.13 as long as the pair of edges do not belong to a length-4 cycle of $\mathcal{C}_1$.

In the following we address the remaining scenario, by moving one edge of the problematic pair to one of the four matchings Y_0, Y_1, Y_2, Y_3. Let $C = (u, v, x, y)$ be a length-4 cycle in $G[M_1 \cup X_2 \cup X_3]$, and assume that edges $(u, v), (x, y) \in X_2 \cup X_3$, edges $(u, y), (v, x) \in M_1$, and edges $(u, x) \in X_0$ and $(v, y) \in X_1$.

Since $\mathcal{C}_2$ do not share any edge with $\mathcal{C}_1$, we consider the degrees of vertices u, v, x, y in $G[M_1 \cup Y_i]$ for $i = 0, 1, 2, 3$. If in one of these four acyclic 2-matchings, say $G[M_1 \cup Y_0]$, at least three of the four vertices have degree 1, say u, v, x, then we can move edge (u, v) from $X_2 \cup X_3$ to Y_0, and thus the problematic pair is resolved. In the other cases, in each $G[M_1 \cup Y_i]$ for $i = 0, 1, 2, 3$, exactly two of the four vertices have degree 1.

Let the two edges of $\mathcal{C}_2$ incident at u (v, x, y, respectively) be $(u, 1)$ and $(u, 1')$ ($(v, 2)$ and $(v, 2')$, $(x, 3)$ and $(x, 3')$, $(y, 4)$ and $(y, 4')$, respectively).

If $(u, 1), (y, 4) \in Y_0$, then u and y both have degree 1 in one of $G[M_1 \cup Y_i]$ for $i = 1, 2, 3$, say in $G[M_1 \cup Y_3]$. It follows that if the other edge of $\mathcal{C}_2$ incident at vertex 1 does not belong to Y_3, then we can move edge $(u, 1)$ from Y_0 to Y_3, and move edge (u, v) from $X_2 \cup X_3$ to Y_0 to resolve the problematic pair; or if the other edge of $\mathcal{C}_2$ incident at vertex 4 does not belong to Y_3, then we can move edge $(y, 4)$ from Y_0 to Y_3, and move edge (x, y) from $X_2 \cup X_3$ to Y_0 to resolve the problematic pair. In the remaining scenario, the other edge of $\mathcal{C}_2$ incident at vertex 1 (vertex 4, respectively) belongs to Y_3. Note that in either $G[M_1 \cup Y_1]$ or $G[M_1 \cup Y_2]$, vertex u has degree 1, and we assume without loss of generality that vertex u has degree 1 in $G[M_1 \cup Y_1]$. Note also that vertex 1 has degree 1 in $G[M_1 \cup Y_1]$. If edge $(y, 4') \notin Y_1$, then vertex y has degree 1 as well, and thus we can move edge $(u, 1)$ from Y_0 to Y_1, and move edge (u, v) from $X_2 \cup X_3$ to Y_0 to resolve the problematic pair; if edge $(y, 4') \in Y_1$ but the other edge of $\mathcal{C}_2$ incident at vertex $4'$ does not belong to Y_3, then we can move edge $(y, 4')$ from Y_1 to Y_3, move edge $(u, 1)$ from Y_0 to Y_1, and move edge (u, v) from $X_2 \cup X_3$ to Y_0 to resolve the problematic pair. Therefore, we only need to argue the scenario where the other edge of $\mathcal{C}_2$ incident at vertex $4'$ belongs to Y_3. Symmetrically considering Y_2, we may assume without loss of generality that the other edge of $\mathcal{C}_2$ incident at vertex $1'$ belongs to Y_3. Consequently, vertices $u, 1, 1'$ all have degree 1 in $G[M_1 \cup Y_1]$, and thus u and at least one of 1 and $1'$ are not connected. If u and 1 are not connected, we can move edge $(u, 1)$ from Y_0 to Y_1, and move edge (u, v) from $X_2 \cup X_3$ to Y_0 to resolve the problematic pair; if u and $1'$ are not connected, we can move edge $(u, 1')$ from Y_2 to Y_1, move edge $(u, 1)$ from Y_0 to Y_2, and move edge (u, v) from $X_2 \cup X_3$ to Y_0 to resolve the problematic pair.

If $(u, 1), (v, 2) \in Y_0$, then u and v both have degree 1 in one of $G[M_1 \cup Y_i]$ for $i = 1, 2, 3$, say in $G[M_1 \cup Y_3]$. The following discussion is very similar to the above paragraph, though slightly simpler. Firstly, if x and y are not connected in $G[M_1 \cup Y_0]$ (u and v are not connected in $G[M_1 \cup Y_3]$, respectively), then we can move edge (x, y) ((u, v),

respectively) from $X_2 \cup X_3$ to Y_0 (Y_3, respectively) to directly resolve the problematic pair. Secondly, if the other edge of $\mathcal{C}_2$ incident at vertex 1 does not belong to Y_3, then we can move edge $(u, 1)$ from Y_0 to Y_3, and move edge (x, y) from $X_2 \cup X_3$ to Y_0 to resolve the problematic pair; or if the other edge of $\mathcal{C}_2$ incident at vertex 2 does not belong to Y_3, then we can move edge $(v, 2)$ from Y_0 to Y_3, and move edge (x, y) from $X_2 \cup X_3$ to Y_0 to resolve the problematic pair. Symmetrically and without loss of generality that $(x, 3), (y, 4) \in Y_3$, if either of the other edges of $\mathcal{C}_2$ incident at vertices 3 and 4 does not belong to Y_3, the problematic pair can be resolved. In the remaining scenario, we assume that vertices u and x have degree 1 in $G[M_1 \cup Y_1]$ (and $(v, 2'), (y, 4') \in Y_1$). Note that vertices $1, 2, 3, 4$ all have degree 1 in $G[M_1 \cup Y_1]$ too. If u and x are not connected in $G[M_1 \cup Y_1]$, then we can swap edges of $X_0 \cup X_1$ and of $X_2 \cup X_3$, and move edge (u, x) from $X_2 \cup X_3$ to Y_1, to resolve the problematic pair. Otherwise, u and 1 should not be connected in $G[M_1 \cup Y_1]$, and we can move edge $(u, 1)$ from Y_0 to Y_1, and move edge (x, y) from $X_2 \cup X_3$ to Y_0, to resolve the problematic pair.

All the other pairs of edges occurring in $\mathcal{C}_2 \cap Y_0$ can be analogously discussed as in either of the above two paragraphs. Repeatedly applying the above process to resolve the problematic pairs of $X_2 \cup X_3$ that reside in a length-4 cycle of $\mathcal{C}_1$, if any. The process moves exactly one edge of the pair from $X_2 \cup X_3$ to either of Y_0, Y_1, Y_2, Y_3, while maintaining $G[M_1 \cup Y_j]$ acyclic for all $j \in \{0, 1, 2, 3\}$. At the end, $G[M_1 \cup X_2 \cup X_3]$ contains no length-4 cycles and therefore $X'_3 = \emptyset$. □

Lemma 2.16. *The weight of matching M_2 is $w'(M_2) \geq \frac{6-\sqrt{2}}{32} w'(\mathcal{C})$.*

Proof. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ denote the two 2-matchings constituting to the maximum weight 4-matching $\mathcal{C}$ of residual graph G' (these two 2-matchings can be obtained from $\mathcal{C}$ in $O(n^{2.5})$ time [37, 49]). Using Lemma 2.10 alone, $\mathcal{C}_1$ can be partitioned into four matchings X_0, X_1, X_2, X_3, and $\mathcal{C}_2$ can be partitioned into four matchings Y_0, Y_1, Y_2, Y_3, respectively, with the desired properties stated in Lemma 2.10. Separately for each partition, we apply Lemmas 2.13 and 2.14 to ensure that it maintains all the the desired properties stated in Lemma 2.10, and it has an extra property that if there is a length-4 cycle in $G[M_1 \cup X_2 \cup X_3]$ (in $G[M_1 \cup Y_2 \cup Y_3]$, respectively), then the two edges of $X_2 \cup X_3$ (of $Y_2 \cup Y_3$, respectively) are from and one is the lightest edge of a length-4 cycle of $\mathcal{C}_1$ (of $\mathcal{C}_2$, respectively).

Recall from Lemmas 2.11 and 2.12 that X'_3 denotes the subset of X_3 of which each edge comes from a length-4 cycle of $G[M_1 \cup X_2 \cup X_3]$ and $X''_3 = X_3 - X'_3$, and X'_2 denotes the subset of X_2 of which each edge comes from a length-4 cycle of $G[M_1 \cup X_2 \cup X_3]$ and $X''_2 = X_2 - X'_2$. We similarly define Y'_3, Y''_3, Y'_2, Y''_2, respectively. From Lemma 2.12,

$w'(X'_3) \le w'(X'_2)$ and $w'(X''_3) \le \frac{1}{2}w'(X''_2)$. Let

$$\delta = \max\left\{\frac{w'(X'_3)}{w'(X''_3)}, \frac{w'(Y'_3)}{w'(Y''_3)}\right\}. \tag{2.15}$$

So we have

$$w'(X_3) = w'(X'_3) + w'(X''_3) \le (1+\delta)w'(X''_3) \le \frac{1+\delta}{2}w'(X''_2) \le \frac{1+\delta}{2}w'(X_2),$$

and similarly

$$w'(Y_3) \le \frac{1+\delta}{2}w'(Y_2).$$

It follows that if $\delta \le \sqrt{2}-1$, we can set M_2 to be the maximum weight matching among the six matchings $\{X_0, X_1, X_2, Y_0, Y_1, Y_2\}$:

$$w'(M_2) \ge \frac{1}{6+(1+\delta)}w'(\mathcal{C}) \ge \frac{6-\sqrt{2}}{32}w'(\mathcal{C}). \tag{2.16}$$

Otherwise, assume without loss of generality that $\frac{w'(X'_3)}{w'(X''_3)} = \delta > \sqrt{2}-1$. We can apply Lemma 2.15 to move exactly one edge from every length-4 cycle of $\mathcal{C}_1$ to either of the four matchings of $\mathcal{C}_2$ such that the new partition of $\mathcal{C}_1$, denoted as W_0, W_1, W_2, W_3, still has all the desired properties described in Lemma 2.10 and has an extra property that $W'_3 = \emptyset$. Note that $W''_3 = X''_3$, and thus

$$w'(X''_3) = w'(W''_3) \le \frac{1}{2}w'(W''_2);$$

$W''_2 = X''_2$ and thus

$$w'(X''_2) = w'(W''_2);$$

$|W'_2| = |X'_2|$ and thus by Lemma 2.14

$$w'(X'_3) \le w'(W'_2).$$

It follows from $w'(X'_3) = \delta w'(X''_3)$ and $W'_3 = \emptyset$ that

$$w'(W_3) = w'(W''_3) = \frac{1}{2+\delta}\left(w'(X'_3) + 2w'(X''_3)\right) \le \frac{1}{2+\delta}\left(w'(W'_2) + w'(W''_2)\right) = \frac{1}{2+\delta}w'(W_2).$$

Therefore, when $\delta > \sqrt{2}-1$, we can set M_2 to be the maximum weight matching among the seven matchings $\{W_0, W_1, W_2, Y_0, Y_1, Y_2, Y_3\}$:

$$w'(M_2) \ge \frac{1}{7+\frac{1}{2+\delta}}w'(\mathcal{C}) \ge \frac{6-\sqrt{2}}{32}w'(\mathcal{C}). \tag{2.17}$$

Equations (2.16) and (2.17) tell that in either case, one matching M_2 can be extracted

from the maximum weight 4-matching $\mathcal{C}$ to extend the first maximum weight matching M_1, and the weight of M_2 is guaranteed to be at least $\frac{6-\sqrt{2}}{32}w'(\mathcal{C}) > 0.1433w'(\mathcal{C})$. □

Lemma 2.17. *The weight of the maximum weight* 4*-matching* $\mathcal{C}$ *is*

$$w'(\mathcal{C}) \geq \max\left\{\frac{1}{2}|B_2|, \frac{1}{4}|B_2| + |B_4|\right\}.$$

Proof. Lemma 2.2 states that the subgraph of induced edges, G'_s, is a 4-matching in graph G' with weight $w'(G'_s) \geq \frac{1}{2}|B_2|$. Therefore, $w'(\mathcal{C}) \geq w'(G'_s) \geq \frac{1}{2}|B_2|$.

On the other hand, graph G' contains all bandpasses of group B_4, and all the edges containing such kind of bandpasses are in the optimal row permutation π^*, which is an acyclic 2-matching in graph G'. Let $w'(\pi^*)$ denote the weight of π^* in graph G', then we have $w'(\pi^*) \geq |B_4|$. From the partitioning scheme for the bandpasses of $S_2(\pi^*)$, we know that no edge in G'_s belongs to π^*. That is, $G'_s \cap \pi^* = \emptyset$. Therefore, in $O(n^{2.5})$ time one can decompose the 4-matching G'_s into two disjoint 2-matchings [37, 49], $\mathcal{C}_1$ and $\mathcal{C}_2$; the unions $\mathcal{C}_1 \cup \pi^*$ and $\mathcal{C}_2 \cup \pi^*$ are both 4-matchings in graph G'. It follows that

$$w'(\mathcal{C}) \geq \frac{1}{2}(w'(\mathcal{C}_1 \cup \pi^*) + w'(\mathcal{C}_2 \cup \pi^*)) = \frac{1}{2}w'(G'_s) + w'(\pi^*)) \geq \frac{1}{4}|B_2| + |B_4|.$$

This proves the lemma. □

Theorem 2.18. *Algorithm* BP3 *is an* $O(n^4)$*-time* 0.5358*-approximation for the bandpass problem.*

Proof. The running time of algorithm BP3 is dominated by the computing for those maximum weight b-matchings, for $b = 1, 2, 4$, which can be done in $O(n^4)$ time. Since M_1 is the maximum weight matching in graph G, from Eq. (2.2) we have

$$w(M_1) \geq \frac{1}{2}p(\pi^*) \geq \frac{1}{2}\left(s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell - 1)\right). \tag{2.18}$$

Combining Eqs. (2.4) and (2.18), we have for any real number $y \in [0, 1]$,

$$w(M_1) \geq \frac{y}{2}\left(s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell - 1)\right) + (1 - y)\left(|B_1| + \frac{1}{2}|B_2| + \frac{2}{3}|B_3|\right). \tag{2.19}$$

The permutation π produced by algorithm BP3 contains $b(\pi) \geq w(M_1) + w'(M_2)$ bandpasses, as indicated at the end of Section 2.2.1. From Lemmas 2.16 and 2.17, we have

$$b(\pi) \geq w(M_1) + w'(M_2) \geq w(M_1) + \frac{6 - \sqrt{2}}{32}\left(\frac{1}{4}|B_2| + |B_4|\right). \tag{2.20}$$

Together with Eqs. (2.3) and (2.19), the above Eq. (2.20) becomes,

$$\begin{aligned}
& b(\pi) \ \geq \ w(M_1) + \frac{6-\sqrt{2}}{32}\left(\frac{1}{4}|B_2| + |B_4|\right) \\
\geq \quad & y\frac{1}{2}\left(s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell-1)\right) \\
& +(1-y)\left(|B_1| + \frac{1}{2}|B_2| + \frac{2}{3}|B_3|\right) + \frac{6-\sqrt{2}}{128}|B_2| + \frac{6-\sqrt{2}}{32}|B_4| \\
= \quad & \frac{y}{2}\left(s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell-1)\right) \\
& +(1-y)|B_1| + \left(\frac{1-y}{2} + \frac{6-\sqrt{2}}{128}\right)|B_2| + \frac{2(1-y)}{3}|B_3| + \frac{6-\sqrt{2}}{32}|B_4| \\
\geq \quad & \frac{46+3\sqrt{2}}{128}\left(s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell-1)\right) + \frac{6-\sqrt{2}}{32}\left(s_2(\pi^*) + \frac{1}{2}|B_1|\right), \quad (2.21)
\end{aligned}$$

where the last inequality is achieved by setting $y = \frac{46+3\sqrt{2}}{64}$. Note that for all $\ell \geq 3$, $(\ell-1) \geq \frac{3}{2}\lfloor\frac{\ell}{2}\rfloor$. It then follows from Eqs. (2.21) and (2.1) that

$$\begin{aligned}
b(\pi) \ &\geq \ \frac{70-\sqrt{2}}{128}\left(s_2(\pi^*) + \frac{46+3\sqrt{2}}{70-\sqrt{2}} \times \frac{3}{2}\sum_{\ell=3}^{n} s_\ell(\pi^*)\left\lfloor\frac{\ell}{2}\right\rfloor\right) \\
&\geq \ \frac{70-\sqrt{2}}{128} b(\pi^*), \quad (2.22)
\end{aligned}$$

where the last inequality holds because $\frac{46+3\sqrt{2}}{70-\sqrt{2}} \times \frac{3}{2} = \frac{138+9\sqrt{2}}{140-2\sqrt{2}} > 1$. That is, the worst-case performance ratio of algorithm BP3 is at least $\frac{70-\sqrt{2}}{128} > 0.5358$. □

2.6 Conclusions and future work

We have presented a series of approximation algorithms BP1, BP2, BP3 for the bandpass problem. Currently, the algorithm BP3 achieves the best approximation ratio $\frac{70-\sqrt{2}}{128} \approx 0.5358$. Our algorithms are based on maximum weight b-matchings, for $b = 1, 2$ and 4, similar to an approach to the closely related Max-TSP. The intrinsic structural property proven for the optimal row permutation and the maximum weight matching is fundamental, without which no better lower bound on the optimum can be built. The partition schemes developed in the literature and our scheme on b-matchings could potentially be further improved. When estimating the performance ratio, the best balance between $|B_2|$ and $|B_4|$ in the second matching M_2 is 1 : 4, which has been achieved in Lemma 2.17. This suggests that future improvements along this line are possible only if one can increase both fractions of $|B_2|$ and $|B_4|$ in the second matching M_2.

Our 4-matching partition scheme produces at least 8 matchings, each of which extends the given matching into an acyclic 2-matching. We also came up with an example (see FIGURE 2.3), where the 4-matching needs to be partitioned into at least 6 matchings such that each matching extends the given matching into an acyclic 2-matching. We

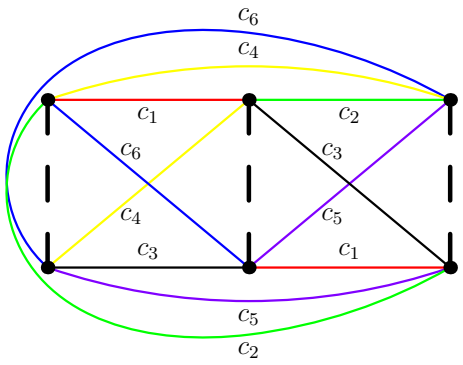

FIGURE 2.3: Each color represents a matching and dashed line means the given matching. This 4-matching needs to be colored with at least 6 colors.

conjecture that the number 6 is "tight", that is, any 4-matching can be partitioned into 6 matchings such that each matching extends the given matching into an acyclic 2-matching. If this is true, we can improve Lemma 2.3, Lemma 2.6, inequality 2.17 immediately and thus there exists an improved approximation algorithm for the *bandpass* problem.

For the Max-TSP problem, Serdyukov presented a $\frac{3}{4}$-approximation algorithm based on the maximum weight *assignment* (called *cycle cover*) and the maximum weight matching [88], which has been improved to the currently best $\frac{7}{9}$-approximation algorithm [78]. We believe that the *bandpass* problem can be better approximated by introducing new structural properties and/or new techniques; yet we also believe that there will be a gap from $\frac{7}{9}$, due to the "dynamic" edge weights. The novel 4-matching partitioning scheme is seemingly better than a similar partition result which is the key to the $\frac{7}{9}$-approximation for the Max-TSP problem. We strongly believe that a better approximation for the Max-TSP problem is possible.

On the other hand, Hassin and Rubinstein gave a randomized approximation algorithm for the Max-TSP problem with expected performance ratio $\frac{25}{33}$ [50] (which was subsequently de-randomized in [27]). It would be interesting to design a randomized approximation for the *bandpass* problem too, with a better-than-0.5358 expected performance ratio.

Chapter 3

Multiple RNA Interaction (MRIP) Problem[1] — An Extension of the Bandpass Problem

3.1 Introduction

RNA interaction is one of the fundamental mechanisms underlying many cellular processes, in particular the genome regulatory code, such as mRNA translation, editing, gene silencing, and synthetic self-assemble RNA design. In the literature, pairwise RNA interaction prediction has been independently formulated as a computational problem, in several works including [3, 66, 79]. While these variants are all motivated by certain biological considerations, the general formulation is usually NP-hard and many special scenarios have been extensively studied [29, 30, 52, 63, 73, 85].

In more complex instances, biologists found multiple small nucleolar RNAs (snoRNAs) interact with ribosomal RNAs (rRNAs) in guiding the methylation of the rRNAs [70], and multiple small nuclear RNAs (snRNA) interact with an mRNA in the splicing of introns [94]. Multiple RNA interactions are believed much more complex than pairwise RNA interactions, where only two RNA molecules are involved. In fact, even if we have a perfect computational framework for pairwise RNA interactions, it might still be difficult to deal with multiple RNA interactions since for a given pool of RNA molecules it is non-trivial to predict their interaction order without sufficient prior biological knowledge.

Motivated by the real needs, Ahmed *et al.* presented in COCOON 2013 their work on multiple RNA interaction prediction, denoted as MRIP [1]. We give some basic definitions to introduce the MRIP problem formally. An RNA molecule is a sequence of nucleotides (A, C, G, and U). A basepair in the RNA is presented as (i, j), where $i < j$, indicating that the i-th nucleotide and the j-th nucleotide form a canonical

[1]This chapter is based on [103, 104].

pairing (*i.e.*, the two nucleotides are either A and U or C and G). The molecule folds into a *structure* which is described as a set of basepairs. In general every nucleotide can participate in at most one basepair, and if not, it is a *free* base (or nucleotide). The set of basepairs is *nested* (a.k.a. secondary structure), if for any two basepairs (i_1, j_1) and (i_2, j_2) with $i_1 < i_2$, either $j_1 < i_2$ or $j_2 < j_1$; otherwise the set is *crossing* (a.k.a. tertiary structure) containing *pseudoknots*. An interaction between two RNAs is a basepair which consists of one free base from each RNA. In the sequel, we use interaction and basepair interchangeably.

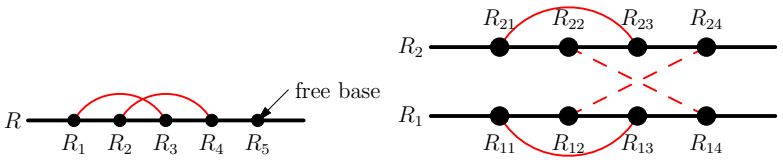

FIGURE 3.1: An illustration of free base, basepair-like structure and pseudoknot-like structure, where the two pairs connected by crossing red dashed lines form to be a pseudoknot-like structure.

In the MRIP problem, we are given a pool of RNAs denoted as $\mathcal{R} = \{R_1, R_2, \ldots, R_m\}$. Without loss of generality, we assume m is even and these RNAs have the same length n. We use $R_{i\ell}$ to denote the ℓ-th base of R_i. Following the formulation by Ahmed *et al.* [1], the possible interactions between every pair of RNAs are assumed known. In fact, these possible interactions can be predicted using existing pairwise RNA interaction predictors [29, 30, 52, 63, 73, 85]. For a possible interaction $(R_{i_1\ell_1}, R_{i_2\ell_2})$, its weight $w(R_{i_1\ell_1}, R_{i_2\ell_2})$ can be set using a probabilistic model or using an energy model or simply at 1 to indicate its contribution to the structure stability. The problem goal is to find out the order of RNAs in which they interact, that the first RNA interacts with the second RNA, which in turn interacts with the third RNA, and so on, and how every two consecutive RNAs interact, so as to maximize the total weight of the interactions (to achieve the most structure stability). Throughout this section, we consider the uni-weight case, that is to maximize the total number of interactions. Two interactions $(R_{i_1\ell_1}, R_{i_2\ell_2})$ and $(R_{i_1k_1}, R_{i_2k_2})$ are pseudoknot-like if $\ell_1 < \ell_2$ but $k_1 > k_2$. The MRIP problem can allow or disallow pseudoknot-like interactions, depending on application details similar to RNA structure prediction.

For a very special case of MRIP (the *Pegs and Rubber Bands* problem in [1]), where the order of interacting RNAs is assumed and pseudoknot-like interactions are disallowed, Ahmed *at al.* proved its NP-hardness and presented a polynomial-time approximation scheme [1]. Given that predicting the interaction order is nontrivial, they also proposed a heuristic for the more general case with unknown interacting order but still disallowing pseudoknot-like interactions.

In this section, we first show that the MRIP allowing pseudoknot-like interactions and with an assumed RNA interaction order can be solved in polynomial time. Secondly, notice that the interactions are basepairs and thus follow the Watson-Crick basepairing rule. For four RNAs $R_{i_1}, R_{i_2}, R_{i_3}, R_{i_4}$, when there are possible interactions $(R_{i_1\ell_1}, R_{i_2\ell_2}), (R_{i_2\ell_2}, R_{i_3\ell_3}), (R_{i_3\ell_3}, R_{i_4\ell_4})$ (for example, they are basepairs (A, U), (U, A), (A, U), respectively), then it is naturally to assume another possible interaction $(R_{i_1\ell_1}, R_{i_4\ell_4})$ between RNAs R_{i_1} and R_{i_4}. If the given interactions satisfy the above property then the MRIP problem is said to have "*transitivity*" property. By looking deep into the MRIP problem which allows transitivity, it is similar to the *bandpass*-2 problem we introduced in the last Chapter. However unlike the *bandpass* problem, the interactions caused by the transitivity are not column-wise in the MRIP problem. We show that the MRIP problem without an assumed RNA interaction order, either allowing or disallowing pseudoknot-like interactions, is NP-hard, and present a constant ratio approximation algorithm for each variant.

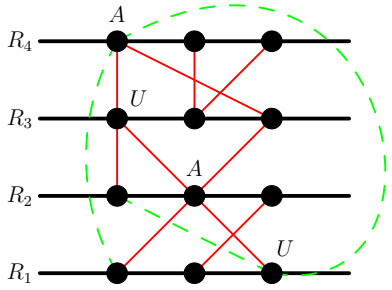

FIGURE 3.2: An illustration of transitivity property, where the pair connected by green dashed line is the possible interaction induced by the transitivity property.

3.2 MRIP with a known RNA interaction order

Within this section, the MRIP problem has a known RNA interaction order, and we assume the order is $(R_1, R_2, \ldots, R_m)$. When disallowing pseudoknot-like interactions, Ahmed *et al.* [1] showed that the problem is NP-hard via a reduction from the *longest common subsequence* problem.

Theorem 3.1. [1] *The MRIP problem disallowing pseudoknot-like interactions is NP-hard.*

When allowing pseudoknot-like interactions, we firstly construct a graph $H = (U, F)$ where every vertex $u_{i\ell}$ corresponds to nucleotide $R_{i\ell}$ and two vertices are connected by an edge if there is a given possible interaction between them. Clearly, one can see

that a matching M of graph H gives a feasible solution to the MRIP problem allowing pseudoknot-like interactions, and vice versa. Therefore, we have the following theorem.

Theorem 3.2. *The* MRIP *problem allowing pseudoknot-like interactions can be solved in polynomial time.*

3.3 The general MRIP

In the general MRIP problem, the RNA interaction order is not assumed. Instead, the possible interactions are given for every pair of RNAs and the problem goal is to find an interaction order achieving the maximum number of interactions.

3.3.1 NP-hardness

Theorem 3.3. *The general* MRIP *problem, either allowing or disallowing pseudoknot-like interactions, is NP-hard.*

Proof. Given a 0-1 matrix $A_{m\times n}$, two consecutive 1's in a column of the matrix is said to form a *bandpass*. When counting the total number of bandpasses in the matrix, no two bandpasses in the same column are allowed to share any common 1. The *Bandpass* problem is to find a row permutation for the input matrix to achieve the maximum total number of bandpasses. Lin proved that the Bandpass problem is NP-hard via a reduction from the *Hamiltonian path* problem [64].

Let the i-th RNA be the i-th row of matrix A, and there is a possible interaction between $R_{i_1\ell_1}$ and $R_{i_2\ell_2}$ if and only if both positions have a 1. Though such constructed RNAs and interactions are not necessarily biologically meaningful, this reduction shows the general MRIP problem is NP-hard. Furthermore, no two possible interactions between a pair of RNAs are *crossing* each other, and thus there are no pseudoknot-like interactions. Hence, the general MRIP problem, either allowing or disallowing pseudoknot-like interactions, is NP-hard. □

3.3.2 A 0.5-approximation algorithm

Using the possible interactions between the pair of RNAs R_i and R_j, we construct a bipartite graph $BG(i,j) = (V_i \cup V_j, E(i,j))$, where the vertex subset V_i (V_j, respectively) corresponds to the set of nucleotides in R_i (R_j, respectively) and the edge set $E(i,j)$ corresponds to the set of given possible interactions between R_i and R_j. That is, if

$(R_{i\ell_1}, R_{j\ell_2})$ is a possible interaction, then there is an edge between $R_{i\ell_1}$ and $R_{j\ell_j}$ in $BG(i,j)$. One clearly sees that when allowing pseudoknot-like interactions, the size of the maximum matching in $BG(i,j)$ is exactly the maximum total number of interactions between RNAs R_i and R_j; when disallowing pseudoknot-like interactions, the maximum total number of interactions between RNAs R_i and R_j can be computed by a dynamic programming algorithm similar to one for computing the longest common subsequence between two given sequences. Either way, this maximum number of interactions is set as the weight between RNAs R_i and R_j, denoted as $w(R_i, R_j)$.

We next construct an edge weighted complete graph G, in which a vertex corresponds to an RNA and the weight between two vertices (RNAs) R_i and R_j is $w(R_i, R_j)$ computed above. Since the optimal solution to the MRIP problem, either allowing or disallowing pseudoknot-like interactions, can be decomposed into two matchings by including alternate edges in the solution, the maximum weight matching M^* of graph G has a weight that is at least half of the total number of interactions in the optimal solution. It follows that this maximum weight matching based algorithm, of which a high-level description is depicted in FIGURE 3.3, is a 0.5-approximation to the MRIP problem.

Input:	m RNAs R_i, $i = 1, 2, \ldots, m$;
Output:	a permutation π of $[m]$ and interactions between RNAs $R_{\pi(i)}$ and $R_{\pi(i+1)}$, for $i = 1, 2, \ldots, m-1$
1.	for each RNA pair R_i and R_j,
	1.1. construct bipartite graph $BG(i,j)$;
	1.2. compute $w(R_i, R_j)$;
2.	construct edge-weighted complete graph G;
3.	compute the maximum weight matching M^* of G;
4.	stack RNA pairs in M^* arbitrarily to form a permutation π;
5.	output π and the interactions in $w(R_{\pi(i)}, R_{\pi(i+1)})$.

FIGURE 3.3: A high-level description of APPROX I.

Theorem 3.4. APPROX I *is a* 0.5*-approximation algorithm for the general MRIP problem, either allowing or disallowing pseudoknot-like interactions.*

Proof. When allowing pseudoknot-like interactions, $w(R_i, R_j)$ can be computed by a maximum matching algorithm in $O(n^3)$ time, where n is the (common) length of the given RNAs; When disallowing pseudoknot-like interactions, $w(R_i, R_j)$ can be computed by a dynamic programming algorithm in $O(n^2)$ time. It follows that the time for constructing graph G is $O(m^2n^3)$. Graph G contains m vertices, and the maximum weight matching M^* can be computed in $O(m^3)$ time. Afterwards, constructing the solution permutation π takes trivially linear time. Therefore, APPROX I is

an $O(\max\{m^3, m^2n^3\})$-time 0.5-approximation algorithm for the MRIP problem allowing pseudoknot-like interactions. For the MRIP problem disallowing pseudoknot-like interactions, its worst-case performance ratio remains 0.5, but its running time is $O(\max\{m^3, m^2n^2\})$. □

3.4 The general MRIP with transitivity

In the last section we proved the NP-hardness for the general MRIP problem, and presented a 0.5-approximation algorithm. One can imagine that the analysis for the 0.5-approximation algorithm must be tight, if the given possible interactions are arbitrary. In this section, we consider a biologically meaningful spectral case where the given possible interactions are *transitive*, that is, for any four RNAs $R_{i_1}, R_{i_2}, R_{i_3}, R_{i_4}$, when there are possible interactions $(R_{i_1\ell_1}, R_{i_2\ell_2}), (R_{i_2\ell_2}, R_{i_3\ell_3}), (R_{i_3\ell_3}, R_{i_4\ell_4})$ (for example, they are basepairs (A, U), (U, A), (A, U), respectively), then $(R_{i_1\ell_1}, R_{i_4\ell_4})$ is also a possible interaction between RNAs R_{i_1} and R_{i_4}. We call it the general MRIP problem with transitivity. Note that in the proof of NP-hardness in Theorem 3.3, the constructed instance of the MRIP problem satisfies the transitivity property. Thus, the general MRIP problem with transitivity, either allowing or disallowing pseudoknot-like interactions, is NP-hard too. We next show that transitivity property can be taken advantage of to design approximation algorithms with performance ratios better than 0.5.

3.4.1 A 0.5328-approximation for disallowing pseudoknots

The improved approximation algorithm for the general MRIP with transitivity and disallowing pseudoknot-like interactions is denoted as Approx II, and its high-level description in provided in Figure 3.4.

Note that in Step 1.2 to compute the maximum number of interactions between two RNAs R_i and R_j while disallowing pseudoknot-like interactions, we can use the same dynamic programming algorithm as used in Approx I, which runs in $O(n^2)$-time. In Step 4.2, the best approximation algorithm for the Maximum-TSP (which has a performance ratio of $\frac{7}{9}$ [78]) is called to compute an acyclic 2-matching; In Step 4.3. to compute a matching M to extend M^*, the union of the edge sets of M and M^*, *i.e.* $G[M \cup M^*]$, is an acyclic 2-matching (sub-tour is another terminology often used in the literature). So basically algorithm Approx II adds to the maximum weight matching M^* of graph G a subset of edges that contains a proven fraction of interactions.

Let I denote the set of interactions in the optimal solution. Let J be set of interactions extracted from the weights of the edges in the maximum weight matching M^* of graph

Input:	m RNAs R_i, $i = 1, 2, \ldots, m$, with transitivity;
Output:	a permutation π of $[m]$ and interactions between RNAs $R_{\pi(i)}$ and $R_{\pi(i+1)}$, for $i = 1, 2, \ldots, m-1$

1. for each RNA pair R_i and R_j,
   - 1.1. construct bipartite graph $BG(i, j)$;
   - 1.2. compute $w(R_i, R_j)$ disallowing pseudoknot-like interactions;
2. construct edge-weighted complete graph G using edge weight function w;
3. compute the maximum weight matching M^* of G;
   - 3.1. delete nucleotides involved in the interactions of M^*;
   - 3.2. reconstruct bipartite graph $BG(i, j)$;
   - 3.3. compute $w'(R_i, R_j)$ disallowing pseudoknot-like interactions;
4. construct edge-weighted complete graph G' using edge weight function w';
   - 4.1. compute the maximum weight 4-matching $\mathcal{C}$ of G';
   - 4.2. compute an approximate acyclic 2-matching $\mathcal{P}$ of G';
   - 4.3. compute a matching M out of $\mathcal{C}$ and $\mathcal{P}$ to extend M^*;
5. stack RNA paths in $G[M^* \cup M]$ arbitrarily to form a permutation π;
6. output π and the interactions in $w(R_{\pi(i)}, R_{\pi(i+1)}) + w'(R_{\pi(i)}, R_{\pi(i+1)})$.

FIGURE 3.4: A high-level description of APPROX II.

G. Note that neither I or J contains crossing interactions. Similarly as in the MRIP problem with a known RNA interaction order (Section 3.2), we construct another graph $H = (U, F)$ for the instance where every vertex $u_{i\ell}$ corresponds to nucleotide $R_{i\ell}$ and two vertices are connected by an edge if there is a given possible interaction between them. With respect to graph H, both I and J are non-crossing matchings. Therefore, the subgraph of H induced by the interactions of I and J, $H[I \cup J]$, is a 2-matching of graph H, denoted by T. Using this 2-matching T, we partition I into 4 subsets of interactions, $I = I_1 \cup I_2 \cup I_3 \cup I_4$, and at the same time partition J into 4 subsets of interactions, $J = J_1 \cup J_2 \cup J_3 \cup J_4$.

Since T is a 2-matching, there are only alternating paths and cycles in T. First we consider paths. For a path of length 1, say $P = \langle u_1, u_2 \rangle$, if its only edge/interaction is in $I \cap J$, then the edge belongs to I_1 and belongs to J_1; if the edge is in $I - J$, then the edge belongs to I_4; if the edge is in $J - I$, then the edge belongs to J_4. For a path of length 3, say $P = \langle u_1, u_2, u_3, u_4 \rangle$, if $(u_1, u_2), (u_3, u_4) \in I$, then they belong to I_2 and edge (u_2, u_3) belongs to J_2. For a path other than the above cases, the edges of I all belong to I_3 and the edges of J all belong to J_3. Afterwards, we consider cycles. For each cycle, the edges of I all belong to I_3 and the edges of J all belong to J_3.

Lemma 3.5. *Let $|X_i|$ denote the size of, that is the number of interactions in, set X_i, for $X = I, J$ and $i = 1, 2, 3, 4$. We have $|J_1| = |I_1|$, $|J_2| = \frac{1}{2}|I_2|$, and $|J_3| \geq \frac{2}{3}|I_3|$.*

Proof. By the definition of I_1, J_1, I_2, J_2, we can easily see $|J_1| = |I_1|$ and $|J_2| = \frac{1}{2}|I_2|$. For I_3 and J_3, from each path or cycle, the number of edges assigned to J_3 is either

greater than or equal to the number of edges assigned to I_3, or 1 less but in this case the length of the path must be at least 5. Therefore, the worst case happens when two and three edges are assigned to J_3 and I_3 respectively, which implies $|J_3| \geq \frac{2}{3}|I_3|$. □

Corollary 3.6. *We have*

$$|I| = |I_1| + |I_2| + |I_3| + |I_4|, \tag{3.1}$$

$$w(M^*) = |J_1| + |J_2| + |J_3| + |J_4|, \tag{3.2}$$

$$w(M^*) \geq |I_1| + \frac{1}{2}|I_2| + \frac{2}{3}|I_3|, \tag{3.3}$$

$$w(M^*) \geq \frac{1}{2}|I| = \frac{1}{2}\left(|I_1| + |I_2| + |I_3| + |I_4|\right). \tag{3.4}$$

Proof. The first two equations are straightforward, following the description of partitioning process and that $w(M^*) = |J|$. The last two inequalities follow from Lemma 3.5 and Theorem 3.4, respectively. □

After deleting bases involved in the interactions of the maximum weight matching M^*, graph G' is constructed the same as graph G except using weight function w'. For a path of length 3 $P = \langle u_1, u_2, u_3, u_4\rangle$, such that $(u_1, u_2), (u_3, u_4) \in I_2$, the transitivity property ensures that there is a possible interaction between u_1 and u_4. Clearly, this interaction is left in graph G', and such an interaction is called an *induced* interaction. Let G'_s be the subgraph of G' that contains exactly those edges each of which is contributed by at least one induced interaction.

Lemma 3.7. *G'_s is a 4-matching in G', and its weight $w'(G'_s) \geq \frac{1}{2}|I_2|$*

Proof. To prove the first part, we only need to prove that every RNA can have induced interactions with at most 4 other RNAs. By the definition of I_2, there is an induced interaction (u_1, u_4) if and only if there is an alternating length-3 path $P = \langle u_1, u_2, u_3, u_4\rangle$, such that $(u_1, u_2), (u_3, u_4) \in I$ and $(u_2, u_3) \in J$. Suppose $u_k \in R_{i_k}$, for $k = 1, 2, 3, 4$. It follows that R_{i_1}, R_{i_2} (R_{i_3}, R_{i_4}, respectively) are adjacent in the optimal permutation and R_{i_2}, R_{i_3} are matched in M^*. Since each RNA can be adjacent to at most two other RNAs in the optimal solution, R_{i_1} and every RNA can have induced interactions with at most 4 other RNAs.

The second part of the lemma follows directly from the definition of an induced interaction, which corresponds to a distinct pair of interactions of I_2. □

It is known that in $O(n^{2.5})$ time, a 4-matching can be decomposed into two 2-matchings [37, 49], and a 2-matching can be further decomposed for our purpose in the next few lemmas.

Lemma 3.8. [28, 98] *Let $\mathcal{C}$ be a 2-matching of graph G such that $M^* \cap \mathcal{C} = \emptyset$. Then, we can partition the edge set of $\mathcal{C}$ into 4 matchings X_0, X_1, X_2, X_3 each of which extends M^*. Moreover, the partitioning takes $O(n\alpha(n))$ time, where $\alpha(n)$ is the inverse Ackerman function.*

The maximum weight 4-matching $\mathcal{C}$ of graph G' can be decomposed into two 2-matchings $\mathcal{C}_1$ and $\mathcal{C}_2$. By Lemma 3.8, $\mathcal{C}_1$ can be partitioned into 4 matchings X_0, X_1, X_2, X_3 and $\mathcal{C}_2$ can be partitioned into 4 matchings Y_0, Y_1, Y_2, Y_3, each of which extends M^*.

Lemma 3.9. [98] *Let $\mathcal{C}$ be a 4-matching of graph G such that $M^* \cap \mathcal{C} = \emptyset$. Then, we can partition the edge set of $\mathcal{C}$ into 8 matchings such that each of them extends M^* and the maximum weight among them is at least $\frac{2}{15}w'(\mathcal{C})$. Moreover, the partitioning takes $O(n^{2.5})$ time.*

Lemma 3.10. *The maximum weight acyclic 2-matching $\mathcal{D}$ of graph G' has weight $w'(\mathcal{D}) \geq |I_4|$.*

Proof. Note that graph G' contains all interactions of I_4 because only bases involved in the interactions of M^* are deleted. The subgraph of graph G' containing exactly the edges contributed by at least one interaction of I_4 is a subgraph of the optimal solution, and thus it is an acyclic 2-matching in graph G'. Therefore,

$$w'(\mathcal{D}) \geq |I_4|.$$

This proves the lemma. □

Lemma 3.11. [28, 98] *Let $\mathcal{P}$ be an acyclic 2-matching of G such that $M^* \cap \mathcal{P} = \emptyset$. Then, we can partition the edge set of $\mathcal{P}$ into three matchings Y_0, Y_1, Y_2 each of which extends M^*. Moreover, the partitioning takes $O(n\alpha(n))$ time.*

Lemma 3.12. [78] *The* Max-TSP *admits an $O(n^3)$-time $\frac{7}{9}$-approximation algorithm, where n is the number of vertices in the graph.*

Corollary 3.13. *The weight of the second matching M to extend M^* has weight $w'(M) \geq \max\left\{\frac{1}{15}|I_2|, \frac{7}{27}|I_4|\right\}$.*

Proof. Using Lemmas 3.7 and 3.9, we have

$$w'(M) \geq \frac{2}{15}w'(\mathcal{C}) \geq \frac{1}{15}|I_2|.$$

Using Lemmas 3.10–3.12, we have

$$w'(M) \geq \frac{1}{3}w'(\mathcal{P}) \geq \frac{7}{27}w'(\mathcal{D}) \geq \frac{7}{27}|I_4|.$$

The corollary holds. □

Theorem 3.14. *Algorithm* Approx II *is a* 0.5328*-approximation for the general MRIP problem with transitivity and disallowing pseudoknot-like interactions.*

Proof. Combining Corollaries 3.6 and 3.13, we have for any real $x, y \in [0, 1]$,

$$\begin{aligned} w(\pi) &= w(M^*) + w'(M) \\ &\geq x(|I_1| + \frac{1}{2}|I_2| + \frac{2}{3}|I_3|) + (1-x)\frac{1}{2}(|I_1| + |I_2| + |I_3| + |I_4|) \\ &\quad + y\frac{1}{15}|I_2| + (1-y)\frac{7}{27}|I_4| \\ &= \frac{1+x}{2}|I_1| + \frac{15+2y}{30}|I_2| + \frac{3+x}{6}|I_3| + \frac{41-27x-14y}{54}|I_4| \\ &\geq \frac{255}{426}|I_1| + \frac{227}{426}|I_2| + \frac{227}{426}|I_3| + \frac{227}{426}|I_4| \\ &\geq \frac{227}{426}|I|, \end{aligned}$$

where the second last inequality holds by setting $x = \frac{14}{71}$ and $y = \frac{35}{71}$. □

3.4.2 A 0.5333-approximation for allowing pseudoknots

The improved approximation algorithm for the general MRIP with transitivity and allowing pseudoknot-like interactions is denoted as Approx III, and its high-level description in provided in Figure 3.5.

Input:	m RNAs R_i, $i = 1, 2, \ldots, m$, with transitivity;
Output:	a permutation π of $[m]$ and interactions between RNAs $R_{\pi(i)}$ and $R_{\pi(i+1)}$, for $i = 1, 2, \ldots, m-1$
1.	for each RNA pair R_i and R_j,
	1.1. construct bipartite graph $BG(i, j)$;
	1.2. compute $w(R_i, R_j)$ allowing pseudoknot-like interactions;
2.	construct edge-weighted complete graph G using edge weight function w;
3.	compute the maximum weight matching M^* of G;
	3.1. delete nucleotides involved in the interactions of M^*;
	3.2. reconstruct bipartite graph $BG(i, j)$;
	3.3. compute $w'(R_i, R_j)$ allowing pseudoknot-like interactions;
4.	construct edge-weighted complete graph G' using edge weight function w';
	4.1. compute the maximum weight 4-matching $\mathcal{C}$ of G';
	4.2. compute a matching M out of $\mathcal{C}$ to extend M^*;
5.	stack RNA paths in $G[M^* \cup M]$ arbitrarily to form a permutation π;
6.	output π and the interactions in $w(R_{\pi(i)}, R_{\pi(i+1)}) + w'(R_{\pi(i)}, R_{\pi(i+1)})$.

Figure 3.5: A high-level description of Approx III.

Approx III is very similar to Approx II, and only differs at two places. Firstly, since the problem allows pseudoknot-like interactions, we run a maximum weight bipartite matching algorithm to compute those edge weights, in Steps 1.2 and 3.3. Secondly, computing a matching M to extend M^* is now based only on the maximum weight 4-matching $\mathcal{C}$, of which the weight has a better estimation due to allowing pseudoknot-like interactions.

The analysis of the algorithm flows also very similarly as in the last section. Again we do the exactly the same interaction partitioning for the optimal solution and the maximum weight matching M^*. One can easily verify that Lemma 3.5, Corollary 3.6, and Lemma 3.7 hold. The following lemma is key to the improvement, which estimates a better lower bound on the weight of the maximum weight 4-matching.

Lemma 3.15. *The weight of the maximum weight* 4*-matching* $\mathcal{C}$ *of graph* G' *is*

$$w'(\mathcal{C}) \geq \max\{\frac{1}{2}|I_2|, \frac{1}{4}|I_2| + |I_4|\}. \tag{3.5}$$

Proof. The first component straightly follows from Lemma 3.7 since G'_s is a 4-matching in graph G'. Note also that graph G' contains all the edges of the optimal solution, each of which is contributed by at least one interaction of I_4. This remainder optimal solution, denoted as $\mathcal{P}$, is an acyclic 2-matching in G', and has weight $w'(\mathcal{P}) \geq |I_4|$.

Since G'_s is a 4-matching, it can be decomposed into two 2-matchings denoted as $\mathcal{D}_1$ and $\mathcal{D}_2$. One clearly see that both $\mathcal{P} \cup \mathcal{D}_1$ and $\mathcal{P} \cup \mathcal{D}_2$ are 4-matchings in graph G'. The interactions of I_4 counted towards $\mathcal{P}$ are not counted towards G'_s. Therefore, we have

$$\begin{aligned} w'((C)) &\geq \max\{w'(\mathcal{P} \cup \mathcal{D}_1), w'(\mathcal{P} \cup \mathcal{D}_2)\} \\ &\geq \frac{1}{2}(w'(\mathcal{D}_1) + w'(\mathcal{D}_2)) + w'(\mathcal{P}) \\ &= \frac{1}{2}w'(G'_s) + |I_4| \\ &\geq \frac{1}{4}|I_2| + |I_4|. \end{aligned}$$

This proves the lemma. □

Theorem 3.16. *Algorithm* Approx III *is a* 0.5333*-approximation for the general MRIP problem with transitivity and allowing pseudoknot-like interactions.*

Proof. The estimation of the performance ratio of 0.5333 is very similar to that of ratio 0.5328 in Theorem 3.14, and is omitted from here. □

3.5 Conclusions and future work

We studied the multiple RNA interaction problem. For the general version, we proved its NP-hardness and also proposed a simple 0.5-approximation algorithm. We tend to believe that the general MRIP could be difficult to solve. Thus any inapproximablity proof would be interesting. Based on the conditions whether the transitivity is allowed, whether the pseudoknot-like interactions are allowed and whether the RNA interaction order are assumed, we investigated several biologically meaningful variants of the MRIP. By looking deep into the variants which allow transitivity, they are similar to the *bandpass*-2 problem we introduced in the last Chapter. However unlike the *bandpass* problem, the interactions caused by the transitivity are not column-wise in these variants. We designed several approximation algorithms for these variants. Motivated by the real applications, it would be interesting to study other variants of the multiple RNA interaction problem under other biologically meaningful assumptions.

Chapter 4

Exemplar Non-Breakpoint Similarity (ENBS) Problem[1]

4.1 Introduction

In genome comparison and rearrangement studies, the breakpoint distance is one of the most well-known distance measures [109]. The implicit idea of breakpoints was initiated as early as in 1936 by Sturtevant and Dobzhansky [93]. But until only a few years ago, it had always been assumed that every gene appears in a genome exactly once. Under this assumption, the genome rearrangement problem is essentially the problem of comparing and sorting unsigned (or signed) permutations [45, 48]. Computing the breakpoint distance between two *perfect* genomes, in which every gene appears exactly once, can be done in linear time.

Perfect genomes are hard to obtain and so far, can only be obtained in several small virus genomes. In fact, perfect genomes do not occur on eukaryotic genomes where paralogous genes are common [76, 87]. In practice, it is important to compute genomic distances between perfect genomes, such as is done by using Hannenhalli and Pevzner method [48]. However, one might have to handle the gene duplication problem. In 1999, Sankoff proposed a way to select, from the duplicated copies of a gene, the common ancestral gene such that the distance between the reduced perfect genomes (called *exemplar genomes*) is minimized. For this case, Sankoff produced a branch-and-bound algorithm [87]. In a subsequent work, Nguyen, Tay and Zhang proposed a divide-and-conquer method to compute the exemplar breakpoint distance empirically [76].

From the algorithm complexity research viewpoint, it has been shown that computing the *exemplar signed reversal distance* and computing the *exemplar breakpoint distance* between two *imperfect* genomes are both NP-hard [17]. A few years ago, Blin and Rizzi further proved that computing the *exemplar conserved interval distance* between two imperfect genomes is NP-hard [14]; furthermore, it is NP-hard to compute the *minimum conserved interval matching*, that is, without deleting the duplicated copies

[1]This chapter is based on [25].

of genes. On the approximability, for any exemplar genomic distance measure $d(\cdot, \cdot)$ satisfying coincidence axiom (i.e., $d(G, H) = 0$ if and only if $G = H$ or the reversal of H), it was shown that the problem does not admit any approximation algorithms, even when each gene appears at most three time in each input genome [26, 112] unless P = NP. Slightly later, this bound was tightened, as deciding when $d(\mathcal{G}, \mathcal{H}) = 0$ is NP-complete even if each gene appears in the input genomes $\mathcal{G}$ and $\mathcal{H}$ at most twice [6, 55]. It follows that for the *exemplar breakpoint distance* and the *exemplar conserved interval distance* problems, there are no polynomial time approximation algorithms. Furthermore, even under a weaker definition of polynomial time approximation algorithms, the exemplar breakpoint distance problem is shown not to admit any weak $O(n^{1-\epsilon})$-approximation algorithm, for any $0 < \epsilon < 1$, where n is the maximum length of the two input genomes [26]. The *exemplar conserved interval distance* problem is also shown not to admit any weak $O(n^{1.5})$-approximation algorithm [24, 112].

Complementary to the genomic distances, computing certain genomic similarities between two genomes has also been studied in [19]. In general, genomic similarity measures do not satisfy coincidence axiom. Among others, Chauve *et al.* proved that computing the *maximum exemplar common interval similarity* between two imperfect genomes is NP-hard, while leaving open the problem approximability [19].

Here we study the *non-breakpoint similarity* between two imperfect genomes, which complements the breakpoint distance measure. Formally, given an alphabet Σ of n genes and two imperfect genomes $\mathcal{G}$ and $\mathcal{H}$ drawn from Σ, the *exemplar non-breakpoint similarity* (ENBS) problem is to delete duplicated genes from $\mathcal{G}$ and $\mathcal{H}$ such that the number of non-breakpoints between the two resultant exemplar genomes, G and H, is maximized. The ENBS problem is NP-hard, and here we study the approximability. When one of the input genomes is already exemplar, the problem is called *one-sided* ENBS; the general case is called *two-sided* ENBS. We first present a linear reduction from the *maximum independent set* (MIS) problem to the *one-sided* 2*-repetitive* ENBS problem. This reduction implies that the *one-sided* ENBS problem is $W[1]$-hard, and that it does not admit an $O(n^{0.5-\epsilon})$-approximation algorithm, for any $\epsilon > 0$, unless NP = ZPP. The W[1]-hardness (see [38] for details) and the recent lower bound results [20] imply that, if k is the optimal solution value to the *one-sided* ENBS problem, then barring an unlikely collapse in the parameterized complexity theory, the problem is not solvable in time $f(k)n^{o(k)}$, for any function f. Our second positive result is an $O(n^{0.5})$-approximation for the *two-sided* 2*-repetitive* ENBS problem. Ignoring constants, the negative hardness result and the positive algorithmic result match perfectly.

4.2 Preliminaries

In the (pairwise) genome comparison and rearrangement problems, we are given two genomes, each of which is a sequence of signed (or unsigned) genes. Note that in general a genome can be a set of such sequences; we focus on such one-sequence genomes, often called *singletons*. The order of the genes in one genome corresponds to their physical positions on the genome, and the sign of a gene indicates which one of the two DNA strands the gene is located. In the literature, most of the research assumes that each gene occurs exactly once in a genome; such an assumption is problematic in reality for eukaryotic genomes and the like where duplications of genes exist [87]. For such an imperfect genome, Sankoff proposed to select an *exemplar genome*, by deleting redundant copies of each gene, in which every gene appears exactly once. The deletion is to minimize certain genomic distance between the resultant exemplar genomes [87].

The following definitions are very much the same as those in [17, 26]. Here, we consider only unsigned genomes, though our results can be applied to signed genomes. We assume a gene alphabet Σ that consists of n distinct genes. A genome $\mathcal{G}$ is a sequence of elements of Σ, under the constraint that each element occurs at least once in $\mathcal{G}$. We allow repetitions of every gene in any genome. Specifically, if each gene occurs exactly once in a genome, then the genome is called *perfect* or *exemplar*; otherwise *imperfect*. A genome $\mathcal{G}$ is called *r-repetitive* if each gene occurs at most r times in $\mathcal{G}$. For example, if $\Sigma = \{a, b, c\}$, then genome $\mathcal{G} = abcbaa$ is 3-repetitive.

Given an imperfect genome $\mathcal{G}$, one can delete the redundant copies of all genes to obtain an exemplar sub-genome G, in which each gene from Σ occurs exactly once. For example, if $\Sigma = \{a, b, c\}$ and genome $\mathcal{G} = abcbaa$, then there are four distinct exemplar genomes for $\mathcal{G}$ by deleting two copies of a and one copy of b: $G_1 = abc$, $G_2 = acb$, $G_3 = bca$, and $G_4 = cba$.

For two exemplar genomes G and H drawn from a common n-gene alphabet Σ, a breakpoint in G is a two-gene substring $g_i g_{i+1}$ that, and its reverse $g_{i+1} g_i$, do not occur as a substring in H. The number of breakpoints in G (symmetrically the number of breakpoints in H) is called the *breakpoint distance* between G and H, and denoted as $\mathrm{bd}(G, H)$. For two imperfect genomes $\mathcal{G}$ and $\mathcal{H}$, their *exemplar breakpoint distance* $\mathrm{ebd}(\mathcal{G}, \mathcal{H})$ is the minimum $\mathrm{bd}(G, H)$, where G and H are exemplar genomes of $\mathcal{G}$ and $\mathcal{H}$, respectively.

For two exemplar genomes G and H drawn from a common n-gene alphabet Σ, a *non-breakpoint* (or *adjacency*) in G is a two-gene substring $g_i g_{i+1}$ that, or its reverse $g_{i+1} g_i$, also occurs as a substring in H. Likewise, the number of non-breakpoints in G (symmetrically the number of non-breakpoints in H) is called the *non-breakpoint similarity*

between G and H, and denoted as $\text{nbs}(G, H)$. Mutatis mutandis, for two imperfect genomes $\mathcal{G}$ and $\mathcal{H}$, their *exemplar non-breakpoint similarity* $\text{enbs}(\mathcal{G}, \mathcal{H})$ is the maximum $\text{nbs}(G, H)$, where G and H are exemplar genomes of $\mathcal{G}$ and $\mathcal{H}$, respectively. Clearly, (exemplar, respectively) breakpoint distance and (exemplar, respectively) non-breakpoint similarity are complement to each other, and they sum to exactly $n - 1$.

Formally, in the *exemplar non-breakpoint similarity* (ENBS) problem, we are given two genomes $\mathcal{G}$ and $\mathcal{H}$ drawn from a common n-gene alphabet Σ, and the goal is to compute $\text{enbs}(\mathcal{G}, \mathcal{H})$ and the associated exemplar genomes G and H of $\mathcal{G}$ and $\mathcal{H}$ respectively.

4.3 Inapproximability result

For (any instance of) the ENBS problem, OPT denotes the optimal solution value. We first have the following lemma.

Lemma 4.1. *$0 \leq OPT \leq n - 1$, where n (≥ 4) is the size of the gene alphabet.*

Proof. Let the n (≥ 4) distinct genes be denoted as $1, 2, 3, \ldots, n$. We only consider the exemplar genomes. The upper bound of OPT is achieved by setting $G = H$; the lower bound of OPT is achieved by setting $G = 123\ldots(n-1)n$ (the identity permutation) and H as follows:

$$H = \begin{cases} (n-1)(n-3)\ldots 531n(n-2)\ldots 642, & \text{if } n \text{ is even,} \\ (n-1)(n-3)\ldots 642n135\ldots(n-4)(n-2), & \text{otherwise.} \end{cases}$$

It can be easily confirmed that between this pair of G and H there is no non-breakpoint. □

It is interesting to note that, given $\mathcal{G}$ and $\mathcal{H}$, whether or not $\text{OPT} = 0$ can be easily confirmed in polynomial time. For instance, one can use a brute-force method on each pair of distinct genes to check whether it is possible to make them into a non-breakpoint. Such an observation implies that there is a trivial $O(n)$-approximation algorithm for the ENBS problem. Note that the complement *exemplar breakpoint distance* problem is different, which does not admit any polynomial time approximation at all since deciding whether its optimal solution value is zero is NP-complete [6, 26, 55]. The next theorem shows that the *one-sided* ENBS problem does not admit any $O(n^{0.5-\epsilon})$-approximation algorithm, for any $\epsilon > 0$.

Theorem 4.2. *Even if one of $\mathcal{G}$ and $\mathcal{H}$ is exemplar and the other is 2-repetitive, the ENBS problem does not admit any $O(n^{0.5-\epsilon})$-approximation algorithm, for any $\epsilon > 0$, unless NP = ZPP, where n is the size of the gene alphabet.*

Proof. It is easy to see that the decision version of the ENBS problem is in NP. We next present a reduction from the *maximum independent set* (MIS) problem to the ENBS problem in which the optimal solution value is preserved. The MIS problem is a well known NP-hard problem that cannot be approximated within a factor of $|V|^{1-\epsilon}$, for any $\epsilon > 0$, unless NP = ZPP, where V is the vertex set of the input graph [51].

Let (V, E) be an instance of the MIS problem, where V is the vertex set and E is the edge set. Let $N = |V|$ and $M = |E|$, and the vertices of V are $v_1, v_2, v_3, \ldots, v_N$, the edges of E are $e_1, e_2, e_3, \ldots, e_M$. We construct a gene alphabet Σ and two genomes $\mathcal{G}$ and $\mathcal{H}$ as follows. For each vertex v_i, two distinct genes v_i and v_i' are created; for each edge e_j, three distinct genes e_j, x_j and x_j' are created. The alphabet Σ contains in total $2N + 3M$ distinct genes. Let A_i denote the sequence of all edges incident at vertex v_i, sorted by their indices. Let $Y_i = v_i A_i v_i'$, for $i = 1, 2, \ldots, N$, and $Y_{N+j} = x_j x_j'$, for $j = 1, 2, \ldots, M$.

Let

$$\mathcal{G} = v_1 v_1' v_2 v_2' \ldots v_N v_N' x_1 e_1 x_1' x_2 e_2 x_2' \ldots x_M e_M x_M'.$$

Clearly, $\mathcal{G}$ is exemplar. We distinguish two cases to construct $\mathcal{H}$ (as in the proof of Lemma 4.1):

$$\mathcal{H} = \begin{cases} Y_{N+M-1} Y_{N+M-3} \ldots Y_1 Y_{N+M} Y_{N+M-2} \ldots Y_2, & \text{if } N+M \text{ is even,} \\ Y_{N+M-1} Y_{N+M-3} \ldots Y_2 Y_{N+M} Y_1 Y_3 \ldots Y_{N+M-2}, & \text{otherwise.} \end{cases}$$

Clearly, in either case, $\mathcal{H}$ is 2-repetitive. The remaining argument is identical for both cases.

We claim that graph (V, E) has a maximum independent set of size k iff $\text{enbs}(\mathcal{G}, \mathcal{H}) = k$. First of all, since $\mathcal{G}$ is exemplar, $G = \mathcal{G}$. If graph (V, E) has an independent set of size k, then the claim is trivial. To see this, we construct the exemplar genome H as follows. For all i, if v_i is in the independent set, then we delete A_i from $Y_i = v_i A_i v_i'$. Next, all other redundant edges can be arbitrarily deleted to form H. This way, $v_i v_i'$ is a non-breakpoint between G and H, and thus $\text{enbs}(\mathcal{G}, \mathcal{H}) = k$. On the other hand, if $\text{enbs}(\mathcal{G}, \mathcal{H}) = k$, the first thing to notice is that $Y_j = x_j x_j'$ $(N+1 \leq j \leq N+M)$ cannot give us any non-breakpoint; so the non-breakpoints between G and H must all come from $Y_i = v_i A_i v_i'$ $(1 \leq i \leq N)$, with A_i being deleted to create a non-breakpoint $v_i v_i'$. It follows that there are exactly k such A_i's being deleted. For any two such deleted A_i and A_j, there is no edge between v_i and v_j, for otherwise both copies of the edge would be deleted and consequently H would not be exemplar. Therefore, these vertices form into an independent set in graph (V, E).

The above reduction takes polynomial time. Since $n = |\Sigma| = 2N + 3M \in O(N^2)$ and the MIS problem does not admit any $O(N^{1-\epsilon})$-approximation algorithm, for any $\epsilon > 0$, unless NP = ZPP, our ENBS problem does not admit any $O(n^{0.5-\epsilon})$-approximation algorithm. □

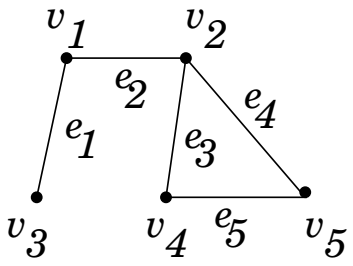

FIGURE 4.1: An illustration of a simple graph for the reduction.

In the example shown in FIGURE 4.1, we have

$\mathcal{G} : v_1v'_1v_2v'_2v_3v'_3v_4v'_4v_5v'_5x_1e_1x'_1x_2e_2x'_2x_3e_3x'_3x_4e_4x'_4x_5e_5x'_5$ and

$\mathcal{H} : x_4x'_4x_2x'_2v_5e_4e_5v'_5v_3e_1v'_3v_1e_1e_2v'_1x_5x'_5x_3x'_3x_1x'_1v_4e_3e_5v'_4v_2e_2e_3e_4v'_2$.

Corresponding to the optimal independent set $\{v_3, v_4\}$, we have
$H : x_4x'_4x_2x'_2v_5e_5v'_5v_3v'_3v_1e_1e_2v'_1x_5x'_5x_3x'_3x_1x'_1v_4v'_4v_2e_3e_4v'_2$. The two non-breaking points are $[v_3v'_3]$ and $[v_4v'_4]$.

4.4 An $O(n^{0.5})$-approximation algorithm

Here we consider the *two-sided* 2*-repetitive* ENBS problem in this section. Let $\Sigma = \{1, 2, \ldots, n\}$ be the gene alphabet, and $\mathcal{G} = (g_1g_2 \ldots g_p)$ and $\mathcal{H} = (h_1h_2 \ldots h_q)$ be the two 2-repetitive genomes. For ease of presentation, for each gene $i \in \Sigma$, an occurrence in $\mathcal{G}$ or its exemplar sub-genomes is denoted by i^+, while an occurrence in $\mathcal{H}$ or its exemplar sub-genomes is denoted by i^-. To implement our algorithm, we construct an interval element y_i^+ between g_i and g_{i+1} for $i = 1, ..., p-1$; likewise, we construct an interval element y_j^- between h_j and h_{j+1} for $j = 1, ..., q-1$. Moreover, for each gene i we construct a gene element x_i.

4.4.1 Algorithm description

Between any two exemplar genomes G and H derived from $\mathcal{G}$ and $\mathcal{H}$ respectively, a non-breakpoint ij un-ambiguously points to two positions i_1 and j_1 in $\mathcal{G}$ and two positions i_2 and j_2 in $\mathcal{H}$ such that $\{g_{i_1}, g_{j_1}\} = \{i^+, j^+\}$ and $\{h_{i_2}, h_{j_2}\} = \{i^-, j^-\}$; furthermore, to obtain G from $\mathcal{G}$, the substring $\mathcal{G}[i_1 + 1..j_1 - 1]$ is deleted (similarly, to obtain H

from $\mathcal{H}$, the substring $\mathcal{H}[i_2+1..j_2-1]$ is deleted). Motivated by this observation, we create a set $S(i_1, j_1; i_2, j_2)$ when $\{g_{i_1}, g_{j_1}\} = \{i^+, j^+\}$ and $\{h_{i_2}, h_{j_2}\} = \{i^-, j^-\}$ for some pair of distinct genes i and j $(i < j)$, for all possible quadruples $1 \leq i_1 < j_1 \leq p$ and $1 \leq i_2 < j_2 \leq q$. Set $S(i_1, j_1; i_2, j_2)$ contains those genes in $\mathcal{G}[i_1+1..j_1-1]$ and those genes in $\mathcal{H}[i_2+1..j_2-1]$, and additionally $j_1 - i_1$ interval elements $y^+_{i_1}, y^+_{i_1+1}, \ldots, y^+_{j_1-1}$, $j_2 - i_2$ interval elements $y^-_{i_2}, y^-_{i_2+1}, \ldots, y^-_{j_2-1}$, and two gene elements x_i and x_j. Clearly, the total number of such constructed sets is $O(n^2)$.

We next remove some of these constructed sets from further consideration, since they do not correspond to feasible non-breakpoints. There are two cases: In one case, $\mathcal{G}[i_1+1..j_1-1]$ contains a gene which occurs only once in $\mathcal{G}$; in the other case, $\mathcal{G}[i_1+1..j_1-1]$ contains both copies of a gene. Since deleting the whole substring $\mathcal{G}[i_1+1..j_1-1]$ of genes leads to no exemplar sub-genomes of $\mathcal{G}$, $g_{i_1}g_{j_1}$ is not a feasible non-breakpoint. The same procedure applies to $\mathcal{H}$, that if $\mathcal{H}[i_2+1..j_2-1]$ contains a gene which occurs only once in $\mathcal{H}$ or contains both copies of a gene, then $h_{i_2}h_{j_2}$ is not a feasible non-breakpoint either. Let $\mathcal{S}$ denote the collection of the constructed sets after the above removing procedure, where each set corresponds to a feasible non-breakpoint.

Let $\Sigma^+ = \{1^+, 2^+, \ldots, n^+\}$, $\Sigma^- = \{1^-, 2^-, \ldots, n^-\}$, $X = \{x_1, x_2, \ldots, x_n\}$, $Y^+ = \{y^+_1, y^+_2, \ldots, y^+_{p-1}\}$, and $Y^- = \{y^-_1, y^-_2, \ldots, y^-_{q-1}\}$. We construct an instance I of *set packing* using the ground set $U = \Sigma^+ \cup \Sigma^- \cup X \cup Y^+ \cup Y^-$ and the collection $\mathcal{S}$ of subsets of U. Then, the linear time (in $|U|$) approximation algorithm in [46] for the *set packing* problem can be applied on I to produce an approximate solution, which is a sub-collection $Approx(I)$ of $\mathcal{S}$ containing mutually disjoint sets. By the following Lemma 4.4, the set of non-breakpoints extracted from $Approx(I)$ can be extended into an exemplar genome G of $\mathcal{G}$ and an exemplar genome H of $\mathcal{H}$, such that $\text{nbs}(G, H) \geq |Approx(I)|$. We return the pair G and H as the final solution to the ENBS problem. A high-level description of the approximation algorithm A is in FIGURE 4.2.

Input: $\Sigma = \{1, 2, \ldots, n\}$ and two 2-repetitive genomes $\mathcal{G}$ and $\mathcal{H}$ Output: Two exemplar genomes G and H of $\mathcal{G}$ and $\mathcal{H}$ respectively
1. Construct set $S(i_1, j_1; i_2, j_2)$ for all possible quadruples; 2. Remove infeasible sets and form set collection $\mathcal{S}$; 3. Construct an instance I of *set packing*: ground set $U = \Sigma^+ \cup \Sigma^- \cup X \cup Y^+ \cup Y^-$ and collection $\mathcal{S}$; 4. Run the linear time *set packing* approximation algorithm on I: obtain a solution $Approx(I)$; 5. Extend $Approx(I)$ into exemplar genomes G and H.

FIGURE 4.2: A high-level description of the approximation algorithm $\mathcal{A}_{ENBS}$.

4.4.2 Performance analysis

We first have the following result for the Set Packing problem:

Lemma 4.3. [46] *The* set packing *problem admits an $O(|U|+|\mathcal{S}|)$-time $|U|^{0.5}$-approximation algorithm, where U is the ground set and $\mathcal{S}$ is the collection of subsets.*

Next, we exploit relationships between feasible solutions of the ENBS and the *set packing* problem.

Lemma 4.4. *If there is a set packing $\mathcal{S}' \subseteq \mathcal{S}$ of size k, then there is a pair of exemplar genomes G and H, derived from $\mathcal{G}$ and $\mathcal{H}$ respectively, such that* $\mathrm{nbs}(G, H) \geq k$.

Proof. Let $S_1, S_2, \ldots, S_k$ be the sets in the set packing $\mathcal{S}'$. Note that these k sets are mutually disjoint, i.e., no two of them contain a common element from $U = \Sigma^+ \cup \Sigma^- \cup X \cup Y^+ \cup Y^-$.

From the construction process of the sets of $\mathcal{S}$, we know each S_i is associated with an interval of $\mathcal{G}$ and an interval of $\mathcal{H}$, and S_i contains all the associated interval elements. Two disjoint sets S_i and S_j are thus associated with two non-overlapping intervals of $\mathcal{G}$ (and of $\mathcal{H}$, respectively). Therefore, all the non-breakpoints corresponding to sets $S_1, S_2, \ldots, S_k$ can be formed by deleting all genes from the intervals associated with sets $S_1, S_2, \ldots, S_k$. Moreover, if a gene i occurs only once in $\mathcal{G}$ (in $\mathcal{H}$, respectively), then i^+ (i^-, respectively) does not belong to any of $S_1, S_2, \ldots, S_k$; likewise, if a gene i occurs twice in $\mathcal{G}$ (in $\mathcal{H}$, respectively), then i^+ (i^-, respectively) belongs to at most one of $S_1, S_2, \ldots, S_k$. Equivalently, gene i either forms into a non-breakpoint together with some other gene, or there is still a copy of it in each of the two genomes after deleting all genes from the intervals associated with sets $S_1, S_2, \ldots, S_k$.

In the former case, element x_i is covered by exactly one of $S_1, S_2, \ldots, S_k$ and thus gene i is in a unique non-breakpoint. In the latter case, we may keep an arbitrary copy of i^+ in $\mathcal{G}$ and an arbitrary copy of i^- in $\mathcal{H}$, while deleting the others if any. This way, we obtain an exemplar genome G from $\mathcal{G}$ and an exemplar genome H from $\mathcal{H}$, for which all the non-breakpoints corresponding to sets $S_1, S_2, \ldots, S_k$ are kept. That is, $\mathrm{nbs}(G, H) \geq k$. This proves the lemma. In addition, we see that such a pair of exemplar genomes can be obtained from $\mathcal{S}'$ in a linear scan through the genomes $\mathcal{G}$ and $\mathcal{H}$. □

Lemma 4.5. *If* $\mathrm{enbs}(\mathcal{G}, \mathcal{H}) = k$, *then the optimal set packing has size at least* $\frac{k}{2}$.

Proof. Let G^* and H^* denote the exemplar genomes of $\mathcal{G}$ and $\mathcal{H}$ respectively such that $\mathrm{nbs}(G^*, H^*) = \mathrm{enbs}(\mathcal{G}, \mathcal{H})$. Clearly, non-breakpoints between G^* and H^*, when

regarded as edges connecting the two involved genes, form non-disjoint paths. For each such path containing ℓ non-breakpoints, a maximum of $\lceil \frac{\ell}{2} \rceil$ disjoint non-breakpoints can be obtained; here two non-breakpoints are disjoint if they do not share any common gene. It follows from the proof of Lemma 4.4 that the optimal set packing has size at least $\frac{k}{2}$. □

Theorem 4.6. *The two-sided* 2*-repetitive* ENBS *problem admits an* $O(n^3)$*-time* $O(n^{0.5})$*-approximation algorithm, where* n *is size of the gene alphabet.*

Proof. Let the two 2-repetitive genomes be $\mathcal{G}$ and $\mathcal{H}$. Their lengths are thus bounded above by $2n$. For each position pair (i_1, j_1) in $\mathcal{G}$, we only need to look up at most 4 possibilities to construct sets, each of which contains $O(n)$ elements. Therefore, the instance I of Set Packing can be constructed in $O(n^3)$ time, with $|U| \leq 7n$ and $|\mathcal{S}| \in O(n^2)$. Running the approximation algorithm for *set packing* on I takes $O(n^2)$ time, with the returned solution $|Approx(I)| \leq n$. Finally, a pair of exemplar genomes G and H can be extended from $Approx(I)$ in $O(n)$ time. Therefore, the overall running time is $O(n^3)$.

From Lemmas 4.3–4.5,

$$\mathrm{nbs}(G, H) \geq |Approx(I)| \geq \frac{\mathrm{enbs}(\mathcal{G}, \mathcal{H})}{2} / |U|^{0.5} = \frac{\mathrm{enbs}(\mathcal{G}, \mathcal{H})}{2\sqrt{7}n^{0.5}}.$$

Therefore, our approximation algorithm has a performance ratio in $O(n^{0.5})$. □

4.5 Conclusions and future work

We studied the exemplar non-breakpoint similarity, complement to the exemplar breakpoint distance, between two imperfect genomes. We proved that the ENBS problem cannot be approximated within $O(n^{0.5-\epsilon})$ for any positive ϵ, even in the *one-sided* 2*-repetitive* case, where n is the size of the gene alphabet. On the positive side, we presented a cubic time $O(n^{0.5})$-approximation algorithm for the *two-sided* 2*-repetitive* ENBS problem. Therefore, within the context of 2-repetitiveness, our negative inapproximability and positive algorithmic results merge perfectly. We believe that our design and analysis techniques could extend the approximation algorithm for r-repetitive, for any fixed r; but we are not sure whether the general ENBS problem admits an $(n^{0.5})$-approximation algorithm, even in the one-sided case. On the other hand, the approximability for the (complement) *one-sided exemplar minimum breakpoint distance* problem, even when each gene appears in the imperfect genome at most twice, is still open. The only known

negative result is APX-hardness [6], and the only positive result is the trivial $O(n)$-factor approximation.

Chapter 5

Minimum Common Integer Partition (MCIP) Problem[1]

5.1 Introduction

The *minimum common integer partition* (MCIP) problem was introduced to the computational biology community by Chen *et al.* [23], formulated from their work on ortholog assignment and DNA fingerprint assembly. Mathematically, a *partition* of a positive integer x is a multiset $\sigma(x) = \{a_1, a_2, \ldots, a_t\}$ of positive integers such that $a_1 + a_2 + \ldots + a_t = x$, where each a_i is called a *part* of the partition of x [4, 5]. For example, $\{3, 2, 2, 1\}$ is a partition of $x = 8$; so is $\{6, 1, 1\}$. A partition of a multiset X of positive integers is the multiset union of the partition $\sigma(x)$ for all x of X, i.e., $\sigma(X) = \uplus_{x \in X} \sigma(x)$. For example, as $\{3, 2, 2, 1\}$ is a partition of $x_1 = 8$ and $\{3, 2\}$ is a partition of $x_2 = 5$, $\{3, 3, 2, 2, 2, 1\}$ is a partition for $X = \{8, 5\}$.

Given a collection of multisets $\{X_1, X_2, \ldots, X_k\}$ ($k \geq 2$), a multiset S is a *common integer partition* (CIP) for them if S is an integer partition of every multiset $X_i, 1 \leq i \leq k$. For example, when $k = 2$ and $X_1 = \{8, 5\}$ and $X_2 = \{6, 4, 3\}$, $\{3, 3, 2, 2, 2, 1\}$ is a CIP for them since $\{3, 3, 2, 2, 2, 1\}$ is also a partition for $X_2 = \{6, 4, 3\}$: $3 + 3 = 6$, $2 + 2 = 4$, and $2 + 1 = 3$. The *minimum common integer partition* (MCIP) problem is defined as to find a CIP for $\{X_1, X_2, \ldots, X_k\}$ with the minimum cardinality. For example, one can verify that, for the above $X_1 = \{8, 5\}$ and $X_2 = \{6, 4, 3\}$, $\{6, 3, 2, 2\}$ is a minimum cardinality CIP. We use k-MCIP to denote the restricted version of the MCIP problem when the number of input multisets is fixed to be k.

For simplicity, we denote the *optimal*, i.e. a minimum cardinality, CIP for $\{X_1, X_2, \ldots, X_k\}$ as $\mathrm{OPT}(X_1, X_2, \ldots, X_k)$, or simply OPT when the input multisets are clear from the context; analogously, we denote the CIP for $\{X_1, X_2, \ldots, X_k\}$ produced by an algorithm A as $\mathrm{CIP}_A(X_1, X_2, \ldots, X_k)$, or simply CIP_A; without the algorithm subscript, we use CIP to denote any feasible common integer partition.

[1] This chapter is based on [105].

We mentioned earlier that the MCIP problem was introduced by Chen *et al.* [23], formulated out of ortholog assignment and DNA fingerprint assembly. The interested readers may refer to their paper for more detailed descriptions and the mappings between the problems. More recently, another application of the MCIP problem in similarity comparison between two unlabeled pedigrees was presented in [54]. Pedigrees, or commonly known as family trees, record the occurrence and appearance (or *phenotypes*) of a particular gene or organism and its ancestors from one generation to the next. They are important to geneticists for linkage analysis, as with a valid pedigree the recombination events can be deduced more accurately [33], or disease loci can be mapped consistently [74, 75]. Jiang *et al.* [54] considered the isomorphism and similarity comparison problems for two-generation pedigrees, and formulated them as the *minimum common integer pair partition* (MCIPP) problem, which generalizes the MCIP problem. By exploiting certain structural properties of the optimal solutions for the 2-MCIP problem, they were able to show that their MCIPP problem is also *fixed-parameter tractable* [54].

5.1.1 Known results

For integer $x \in \mathbb{Z}^+$, its number of integer partitions increases very rapidly with x. For example, integer 3 has three partitions, namely $\{3\}$, $\{2,1\}$, and $\{1,1,1\}$; integer 4 has five partitions, namely $\{4\}$, $\{3,1\}$, $\{2,2\}$, $\{2,1,1\}$, and $\{1,1,1,1\}$; while integer 10 has 190,569,292 partitions according to [4].

Given a collection of multisets $\{X_1, X_2, \ldots, X_k\}$ ($k \geq 2$), they have a CIP if and only if they have the same summation over their elements. Multisets with this property are called *related* [22], and we assume throughout this section that the multisets in any instance of MCIP are related, as the verification takes only linear time.

One can see that the 2-MCIP problem generalizes the well-known *subset sum* problem [34], based on the following observation: given a positive integer number x and a set of positive integers $X = \{a_1, a_2, \ldots, a_m\}$, there exists a subset of X summing to x if and only if for the two multisets $X = \{a_1, a_2, \ldots, a_m\}$ and $Y = \{x, \sum_{i=1}^{m} a_i - x\}$, $|\mathrm{OPT}(X, Y)| = m$. Thus 2-MCIP is NP-hard [22]. Chen *et al.* showed that 2-MCIP is APX-hard [22], via a linear reduction (also called an approximation preserving reduction) from the *maximum bounded* 3*-dimensional matching problem* [56].

Let $M = |X_1| + |X_2| + \ldots + |X_k|$ denote the total number of integers in the k-MCIP problem. For the positive algorithmic results, Chen *et al.* presented a linear time 2-approximation algorithm and an $O(M^9)$-time 5/4-approximation algorithm for 2-MCIP [22], based on a heuristic for the *maximum weighted set packing problem* [56]. The 5/4-approximation can be taken as a subroutine to design a $0.625k$-approximation

algorithm for k-MCIP (when k is even; when k is odd, the approximation ratio is $0.625k+0.375$) [111]. Woodruff developed a framework for capturing the frequencies of the integers across the input multisets and presented a randomized $O(M \log k)$-time approximation algorithm for k-MCIP, with worst-case performance ratio $0.6139k(1 + o(1))$ [111]. The basic idea is, when there are not too many distinct integers in the input multisets, most of the low frequency integers will have to be split into at least two parts in any common partition. Inspired by this idea, Zhao *et. al.* [113] formulated the k-MCIP problem into a *flow decomposition* problem in an acyclic k-layer network with the goal to find a minimum number of directed simple paths from the source to the sink. Since this minimum number can be bounded by the number of arcs in the network according to the well-known *flow decomposition theorem* [2], Zhao *et. al.* presented a scheme to reduce the number of arcs in the network, resulting in a de-randomized approximation algorithm with performance ratio $0.5625k(1 + o(1))$, which is the currently best.

5.1.2 Our contributions

We present a polynomial-time 6/5-approximation algorithm for 2-MCIP. Subsequently, we obtain a $0.6k$-approximation algorithm for k-MCIP when k is even (when k is odd, the approximation ratio is $0.6k + 0.4$). It is worth pointing out that the ratio of $0.5625k$ in [113] is asymptotic, that it holds for only sufficiently large k; while our ratio of $0.6k$ is absolute, that it holds for all $k \geq 2$.

5.2 A 6/5-approximation algorithm for 2-MCIP

In this section, we deal with the 2-MCIP problem. For ease of presentation, we denote the two multisets of positive integers in an instance as $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$, and assume without loss of generality that they are related. Recall that, $\mathrm{OPT}(X, Y)$ denotes the optimal solution — the minimum cardinality CIP for $\{X, Y\}$, and $\mathrm{CIP}_A(X, Y)$ denotes the solution CIP produced by algorithm A.

5.2.1 Preliminaries

Chen *et al.* presented a simple linear time 2-approximation algorithm for 2-MCIP [21, 22], denoted as Apx21. Each iteration of Apx21 chooses an (arbitrary) element $x \in X$ and an (arbitrary) element $y \in Y$, and adds $\min\{x, y\}$ to the solution $\mathrm{CIP}_{\mathrm{Apx21}}$; subsequently, if $x = y$ then x is removed from X and y is removed from Y; otherwise $\min\{x, y\}$ is removed from the multiset it appears in and $\max\{x, y\}$ is replaced with

$\max\{x, y\} - \min\{x, y\}$ in the other multiset. Its performance ratio of 2 is seen from the fact that $|\mathrm{OPT}(X, Y)| \geq \max\{m, n\}$ and that the solution $\mathrm{CIP}_{\mathrm{APX21}}$ contains no more than $m + n - 1$ integers. Consequently, we have the following lemma.

Lemma 5.1. [21, 22] $\max\{m, n\} \leq |\mathrm{OPT}(X, Y)| \leq |\mathrm{CIP}_{\mathrm{APX21}}| \leq m + n - 1$.

Given an instance $\{X, Y\}$ of 2-MCIP and an arbitrary CIP that specifies the integer partitions for all elements of X and Y, we say that $x_i \in X$ is *mapped* to $y_j \in Y$ if there exists an element of CIP that is a part of the partition for x_i and is also a part of the partition for y_j. This mapping relationship gives rise naturally to a bipartite graph $G(X, Y)$, in which the two disjoint subsets of vertices are X and Y, respectively, and vertex x_i and vertex y_j are adjacent if and only if x_i is mapped to y_j according to the CIP. Note that an edge of the bipartite graph G one-to-one corresponds to an element of CIP, and in general there could be multiple edges between a pair of vertices in $G(X, Y)$. In the sequel, we use integer x_i and vertex x_i interchangeably, and use an edge of G and an element of CIP interchangeably.

For a connected component of the bipartite graph $G(X, Y)$, let X' denote its subset of vertices in X and Y' denote its subset of vertices in Y, respectively; then X' and Y' are related and they are called a pair of *related sub-multisets* of X and Y; furthermore, the edges in this connected component form a common integer partition for X' and Y' and denoted as $\mathrm{CIP}(X', Y')$, with $|\mathrm{CIP}(X', Y')| \geq |X'| + |Y'| - 1$.

It might happen that the induced bipartite graph $G(X, Y)$ by any CIP of $\{X, Y\}$ is connected, or equivalently speaking X and Y has no pair of related *proper* sub-multisets. In this case X and Y are *basic* related multisets. For example, $X = \{3, 3, 4\}$ and $Y = \{6, 2, 2\}$ are not basic since $\{3, 3\}$ and $\{6\}$ is a pair of related proper sub-multisets; while $X = \{1, 4\}$ and $Y = \{2, 3\}$ are basic. Define the size of a pair of related multisets X and Y to be the total number of elements in the two multisets, *i.e.* $|X| + |Y|$.

Lemma 5.2. [21, 22] *If X and Y are a pair of basic related multisets, then* $|\mathrm{OPT}(X, Y)| = |X| + |Y| - 1$.

If the minimum size of any pair of related sub-multisets of X and Y is c, then $|\mathrm{OPT}(X, Y)| \geq \frac{c-1}{c}(|X| + |Y|)$.

It is not hard to see that for any instance of 2-MCIP, the bipartite graph corresponding to the optimal solution is a forest of the maximum number of trees, each of which corresponds to a pair of basic related multisets. The main idea in our algorithm is to produce a solution containing as many trees as possible, via packing as many pairs of (basic) related multisets as possible. In the sequel, a set containing a single element

is also denoted by the element, when there is no ambiguity, and $X - X'$ is the set minus/subtraction operation. The next lemma handles size-2 related sub-multisets.

Lemma 5.3. [21, 22] *For an instance* $\{X, Y\}$ *of* 2-MCIP, *if* $x_i = y_j$ *for some* $x_i \in X$ *and* $y_j \in Y$, *then* $x_i \uplus \mathrm{OPT}(X - x_i, Y - y_j)$ *is a minimum* CIP *for* X *and* Y, *i.e.,* $|\mathrm{OPT}(X, Y)| = |\mathrm{OPT}(X - x_i, Y - y_j)| + 1$.

5.2.2 Algorithm description

In this section we present a new approximation algorithm, denoted as Apx65, for computing a CIP for the given two related multisets X and Y. The running time and worst-case performance analyses are done in the next section. Essentially, algorithm Apx65 extends the set packing idea in the 5/4-approximation algorithm [21, 22], to pack well the pairs of basic related sub-multisets of sizes $3, 4$, and 5. Nonetheless, our set packing process is different from the process in the 5/4-approximation algorithm, and the performance analysis is built on several new properties we uncover between $\mathrm{OPT}(X, Y)$ and our $\mathrm{CIP}_{\text{Apx65}}$.

Let $Z = X \cap Y$ denote the sub-multiset of common elements of X and Y. By Lemma 5.3 we know that $\mathrm{OPT}(X - Z, Y - Z) \uplus Z$ is an optimal CIP for X and Y. Therefore, in the sequel we assume without loss of generality that X and Y do not share any common integer. In the first step of algorithm Apx65, all pairs of basic related sub-multisets of X and Y of sizes $3, 4$, and 5 are identified. A pair of basic related sub-multisets of size i is called an i-set, for $i = 3, 4, 5$; the weight $w(\cdot)$ of a 3-set ($4, 5$-set, respectively) is set to 3 ($2, 1$, respectively). We use $\mathcal{C}$ to denote this collection of i-sets for $i = 3, 4, 5$.

Let the ground multiset U contain all elements of X and Y that appear in the i-sets of $\mathcal{C}$. In the second step, the algorithm is to find a set packing of large weight for the *Weighted Set Packing* [12] instance $(U, \mathcal{C})$. To do so, a graph H is constructed in which a vertex one-to-one corresponds to an i-set of $\mathcal{C}$ and two vertices are adjacent if and only if the two corresponding i-sets intersect. This step of computing a *heavy* set packing is iterative [12], denoted by Greedy: suppose P is the current set packing (equivalently an independent set in H, which was initialized to contain all isolated vertices of H), and let $w^2(P) = \sum_{p \in P} w^2(p)$ be the sum of squared weights of all i-sets of P; an independent set T of H (equivalently a sub-collection of disjoint i-sets of $\mathcal{C}$) *improves* $w^2(P)$ if $w^2(T) > w^2(N(T, P))$, where $N(T, P)$ denotes the closed neighborhood of T in P; finally, if there is an independent set T of size ≤ 37 which improves $w^2(P)$, then P is replaced by $(P - N(T, P)) \cup T$; otherwise, the process terminates and returns the current P as the solution set packing.

Input:	Related multisets X and Y.
Output:	A common integer partition $\text{CIP}_{\text{APX65}}$ of X and Y.
1.	1.1. Let $Z = X \cap Y$;
	1.2. $X \leftarrow X - Z$, $Y \leftarrow Y - Z$;
	1.3. Identify $\mathcal{C}$ of all basic related sub-multisets of sizes $3, 4, 5$;
2.	2.1. Let U be the ground multiset;
	2.2. Compute a heavy set packing P for instance $(U, \mathcal{C})$ by GREEDY;
3.	3.1. Let X' and Y' be the sub-multisets of elements covered by P;
	3.2. Run APX21 to compute $\text{CIP}_{\text{APX21}}(X - X', Y - Y')$;
	3.3. Return $Z \uplus \left(\biguplus_{X_0 \uplus Y_0 \in P} \text{OPT}(X_0, Y_0)\right) \uplus \text{CIP}_{\text{APX21}}(X - X', Y - Y')$.

FIGURE 5.1: A high-level description of algorithm APX65.

Let P denote the set packing computed in the second step, and X' and Y' denote the sub-multisets of X and Y, respectively, of which the elements are "covered" by the i-sets of P. Note that P is maximal, in the sense that no more i-set of $\mathcal{C}$ can be appended to P. Therefore, in the remainder 2-MCIP instance $(X - X', Y - Y')$, the minimum size of any pair of related sub-multisets of $X - X'$ and $Y - Y'$ is at least 6. In the last step, algorithm APX21 is run on instance $(X - X', Y - Y')$ to output a solution $\text{CIP}_{\text{APX21}}(X - X', Y - Y')$; the final solution $\text{CIP}_{\text{APX65}}(X, Y)$ is

$$Z \uplus \left(\biguplus_{X_0 \uplus Y_0 \in P} \text{OPT}(X_0, Y_0) \right) \uplus \text{CIP}_{\text{APX21}}(X - X', Y - Y'), \tag{5.1}$$

where $X_0 \uplus Y_0 \in P$ is an i-set in the computed set packing P. A high-level description of algorithm APX65 is depicted in FIGURE 5.1.

5.2.3 Performance analysis

The key to the performance guarantee is to analyze the quality of the computed set packing P in the second step of the algorithm. Let P_i denote the collection of i-sets in P, for $i = 3, 4, 5$, respectively. For the weighted set packing instance $(U, \mathcal{C})$, we consider one optimal set packing Q^* and let Q_i^* denote the sub-collection of i-sets in Q^*, for $i = 3, 4, 5$, respectively. Let $p_i = |P_i|$ and $q_i^* = |Q_i^*|$, for $i = 3, 4, 5$.

We further let Q_{ij}^* be the sub-collection of Q_i^*, each i-set of which intersects with exactly j sets of the computed set packing P, for $i = 3, 4, 5$ and $j = 1, 2, \ldots, i$. Let $q_{ij}^* = |Q_{ij}^*|$. Because the set packing P is maximal, each set of Q^* must intersect with certain set(s) in P. This implies

$$q_i^* = \sum_{j=1}^{i} q_{ij}^*, \ i = 3, 4, 5. \tag{5.2}$$

On the other hand, every i-set of Q^*_{ij} intersects with exactly j sets of P; therefore

$$\sum_{i=3}^{5}\sum_{j=1}^{i}(j \times q^*_{ij}) \leq |X'| + |Y'| = \sum_{i=3}^{5}(i \times p_i). \tag{5.3}$$

Eq. (5.2) and Eq. (5.3) together give

$$\begin{aligned}
& 3q^*_3 + 2q^*_4 + q^*_5 \\
= \;& 3\sum_{j=1}^{3} q^*_{3j} + 2\sum_{j=1}^{4} q^*_{4j} + \sum_{j=1}^{5} q^*_{5j} \\
\leq \;& \left(\sum_{j=1}^{3} jq^*_{3j} + 2q^*_{31} + q^*_{32}\right) + \left(\sum_{j=1}^{4} jq^*_{4j} + q^*_{41}\right) + \left(\sum_{j=1}^{5} jq^*_{5j}\right) \\
= \;& \left(\sum_{i=3}^{5}\sum_{j=1}^{i} jq^*_{ij}\right) + 2q^*_{31} + q^*_{32} + q^*_{41} \\
\leq \;& (3p_3 + 4p_4 + 5p_5) + 2q^*_{31} + q^*_{32} + q^*_{41}.
\end{aligned} \tag{5.4}$$

The following Lemma 5.4 states a key structural relationship between the computed set packing P and the optimal set packing Q^*. Section 3 is devoted to the proof of this lemma.

Lemma 5.4. $3q^*_3 + 2q^*_4 + q^*_5 \leq 5(p_3 + p_4 + p_5)$.

By Lemma 5.3, we assume that there are no common integer elements between the two input multisets X and Y. Lemma 5.5 presents a quality guarantee on the computed solution $\mathrm{CIP}_{\textsc{Apx65}}(X, Y)$, in terms of the set packing P.

Lemma 5.5. $|\mathrm{CIP}_{\textsc{Apx65}}(X, Y)| \leq m + n - (p_3 + p_4 + p_5 + 1)$.

Proof. Note from the description of algorithm Apx65 in Figure 5.1 that, for every i-set of the computed set packing P, its common integer partition has the minimum size $i-1$, for $i = 3, 4, 5$. That is,

$$\left| \biguplus_{X_0 \uplus Y_0 \in P} \mathrm{OPT}(X_0, Y_0) \right| = 2p_3 + 3p_4 + 4p_5, \tag{5.5}$$

where $X_0 \uplus Y_0 \in P$ is an i-set in the computed set packing P. On the other hand, on the remainder instance $(X - X', Y - Y')$, algorithm Apx21 returns a solution

$$|\mathrm{CIP}_{\textsc{Apx21}}(X - X', Y - Y')| \leq m + n - (3p_3 + 4p_4 + 5p_5) - 1. \tag{5.6}$$

The lemma immediately follows from Eqs. (5.5, 5.6). □

We now estimate $\mathrm{OPT}(X,Y)$. Let Q'_i, for $i = 3,4,5$ be the collection of pairs of basic related multisets of size i induced by $\mathrm{OPT}(X,Y)$, and let $q'_i = |Q'_i|$. It is clear that

$$3q'_3 + 2q'_4 + q'_5 \leq 3q^*_3 + 2q^*_4 + q^*_5 \tag{5.7}$$

because Q^* is the maximum weight set packing of the instance $(U,\mathcal{C})$ and certainly $Q = Q'_3 \cup Q'_4 \cup Q'_5$ is also a set packing.

Lemma 5.6. $|\mathrm{OPT}(X,Y)| \geq \frac{5}{6}(m+n) - \frac{1}{6}(3q^*_3 + 2q^*_4 + q^*_5)$.

Proof. Note that for every i-set of the set packing Q, its common integer partition has the minimum size $i-1$, for $i = 3,4,5$. Every other connected component in graph $G(X,Y)$ induced by $\mathrm{OPT}(X,Y)$ has size at least 6. Therefore, by Lemma 5.2 we have

$$\begin{aligned}
|\mathrm{OPT}(X,Y)| &\geq 2q'_3 + 3q'_4 + 4q'_5 + \frac{5}{6}\left(m+n-3q'_3-4q'_4-5q'_5\right) \\
&= \frac{5}{6}(m+n) - \frac{1}{6}(3q'_3 + 2q'_4 + q'_5) \\
&\geq \frac{5}{6}(m+n) - \frac{1}{6}(3q^*_3 + 2q^*_4 + q^*_5).
\end{aligned}$$

This proves the lemma. □

Theorem 5.7. *Algorithm* Apx65 *is a* $\frac{6}{5}$*-approximation for* 2-MCIP.

Proof. We first examine the time complexity of algorithm Apx65. From the description of algorithm Apx65 in Fig 5.1, steps 1.1 and 1.2 can be done in $O(m+n)$ and step 1.3 takes $O((m+n)^5)$ time as there are at most $O((m+n)^5)$ sets in $\mathcal{C}$. Our weighting scheme ensures that each iteration of Greedy increases the sum of squared weights by at least 1. Note that the sum of squared weights of any set packing is upper bounded by $3(m+n)$. We conclude that the total number of iterations in Greedy is $O(m+n)$. In each iteration, we check every sub-collection of $\mathcal{C}$ of size ≤ 37, which takes $O((m+n)^{5\times 37}) = O((m+n)^{185})$ time. That is, step 2 costs $O((m+n)^{186})$ time. Step 3 takes linear time as algorithm Apx21 runs in linear time. In summary, the total running time of our algorithm Apx65 is $O((m+n)^{186})$.

For its worst case performance ratio, by Lemmas 5.4, 5.5 and 5.6, we have

$$\begin{aligned}
\frac{\mathrm{CIP}_{\mathrm{Apx65}}(X,Y)}{\mathrm{OPT}(X,Y)} &\leq \frac{m+n-(p_3+p_4+p_5+1)}{\frac{5}{6}(m+n) - \frac{1}{6}(3q^*_3 + 2q^*_4 + q^*_5)} \\
&\leq \frac{6}{5} \times \frac{m+n-(p_3+p_4+p_5+1)}{m+n-\frac{1}{5}(3q^*_3 + 2q^*_4 + q^*_5)} \\
&\leq \frac{6}{5} \times \frac{m+n-(p_3+p_4+p_5+1)}{m+n-(p_3+p_4+p_5)}
\end{aligned}$$

$$< \frac{6}{5},$$

where $m = |X|$ and $n = |Y|$. Therefore, APX65 is a $\frac{6}{5}$-approximation. □

5.3 Proof of Lemma 5.4

Recall the termination condition of algorithm GREEDY for computing the heavy set packing P, that is, there is no independent set T such that $|T| \leq 37$ and T improves $w^2(P)$. Also recall the weighting scheme $(w_3, w_4, w_5) = (3, 2, 1)$, where w_i is the weight of an i-set of $\mathcal{C}$. We summarize in the following Lemma 5.8 some useful properties of the sets in Q^*_{31}, Q^*_{41}, and Q^*_{32}, see also FIGURE 5.2. Their proofs are straightforward using the termination condition and the weight scheme, and we skip them.

Lemma 5.8. (a) *Every set of Q^*_{31} intersects with a set of P_3, and no other set of $Q^*_{31} \cup Q^*_{41}$ can intersect with this set of P_3; such sets of P_3 form a sub-collection denoted as P^1_3.*

(b) *Every set of Q^*_{41} intersects with a set of $P_3 \cup P_4$.*

(b1) *If two sets of Q^*_{41} intersect with a common set of P, then this set belongs to P_3, and no other set of Q^*_{41} can intersect with this set of P_3; such sets of P_3 form a sub-collection denoted as P^2_3, and such sets of Q^*_{41} form a sub-collection denoted as Q^{*1}_{41}.*

(b2) *If only one set of Q^*_{41} intersects with a set of P_3, then no other set of $Q^*_{31} \cup Q^*_{41}$ can intersect with this set of P_3; such sets of P_3 form a sub-collection denoted as P^3_3, and such sets of Q^*_{41} form a sub-collection denoted as Q^{*2}_{41}.*

(b3) *Otherwise, a set of Q^*_{41} intersects with a set of P_4, and no other set of $Q^*_{31} \cup Q^*_{41}$ can intersect with this set of P_4; such sets of Q^*_{41} form a sub-collection denoted as Q^{*3}_{41}.*

Let $P^4_3 = P_3 - P^1_3 - P^2_3 - P^3_3$. Clearly, $\{P^1_3, P^2_3, P^3_3, P^4_3\}$ is a partition of P_3; so is $\{Q^{*1}_{41}, Q^{*2}_{41}, Q^{*3}_{41}\}$ a partition of Q^*_{41}.

(c) *Every set of Q^*_{32} must intersect a set of P_3.*

Let $p^j_3 = |P^j_3|$ for $j = 1, 2, 3, 4$, and $q^{*j}_{41} = |Q^{*j}_{41}|$ for $j = 1, 2, 3$.

Lemma 5.9. *We have the following relationships: $p_3 = p^1_3 + p^2_3 + p^3_3 + p^4_3$, $q^*_{41} = q^{*1}_{41} + q^{*2}_{41} + q^{*3}_{41}$, $p^1_3 = q^*_{31}$, $p^2_3 = \frac{1}{2}q^{*1}_{41}$, and $p^3_3 = q^{*2}_{41}$.*

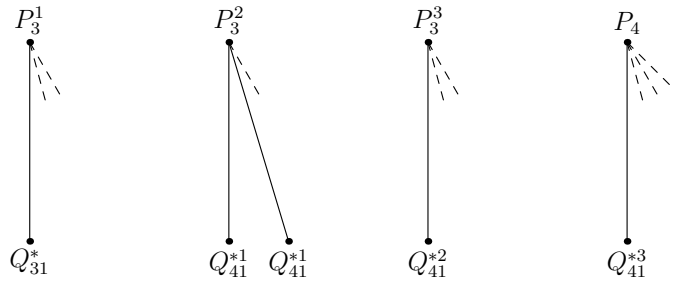

FIGURE 5.2: The definitions of sub-collections of P_3 and $Q_{41}*$ using the set intersecting configurations, where a solid (dashed, respectively) line indicates a firm (possible, respectively) set intersection.

Proof. The first two equalities hold since, by Lemma 5.8(b), $\{P_3^1, P_3^2, P_3^3, P_3^4\}$ is a partition of P_3, and $\{Q_{41}^{*1}, Q_{41}^{*2}, Q_{41}^{*3}\}$ is a partition of Q_{41}^*.

The third equality holds by Lemma 5.8(a) that the sets of Q_{31}^* and the sets of P_3^1 one-to-one correspond to each other. Analogously, the fourth equality holds due to Lemma 5.8(b1) that the sets of Q_{41}^{*1} are paired up, and these pairs and the sets of P_3^2 one-to-one correspond to each other; the fifth equality holds by Lemma 5.8(b2) that the sets of Q_{41}^{*2} and the sets of P_3^3 one-to-one correspond to each other. □

We next construct a bipartite graph H', which is an induced subgraph of graph H that we constructed for the Weighted Set Packing instance $(U, \mathcal{C})$, as follows: One subset of vertices of H' is $Q_{31}^* \cup Q_{32}^* \cup Q_{41}^*$ (which is a sub-collection of the optimal set packing Q^*), and the other subset of vertices of H' is P (which is the computed set packing), and again two vertices are adjacent if and only if the corresponding two sets intersect. In the sequel, we use the set of $\mathcal{C}$ and the vertex of graph H (or H') interchangeably; we also abuse the sub-collection, such as Q_{31}^*, of sets to denote the corresponding vertex subset in graph H (or H'). Once again recall that the termination condition of GREEDY tells that there is no improving sub-collection of $Q_{31}^* \cup Q_{32}^* \cup Q_{41}^*$ of size ≤ 37.

We prove Lemma 5.4 by showing that the inequality holds in every connected component of graph H', followed by a straightforward linear summation over all connected components. We therefore assume without loss of generality that graph H' is connected.

The following lemma says that the set packing algorithm GREEDY packs a lot more 3-sets into P than 4-sets and 5-sets.

Lemma 5.10. *If $p_4 + p_5 \geq 2$, then in graph H' the length of the shortest path between any two vertices $a, b \in P_4 \cup P_5$ is $d(a, b) \geq 76$; consequently, $p_3 \geq 37$ and $p_4 + p_5 \leq \frac{1}{18} p_3$.*

Proof. Let a and b be two vertices of $P_4 \cup P_5$ such that there is no other vertex from $P_4 \cup P_5$ on a shortest path connecting them in graph H'. Since H' is bipartite, this path

has an even length and is denoted as

$$\langle a = a_0, c_0, a_1, c_1, \ldots, a_\ell, c_\ell, a_{\ell+1} = b \rangle,$$

for some $\ell \geq 0$ (see FIGURE 5.3). Since every vertex c_i on the path has degree at least 2, it has to belong to Q_{32}^* and consequently it has degree exactly 2 in graph H'. It follows that for the independent set $T = \{c_0, c_1, \ldots, c_\ell\}$, $w^2(T) = 9(\ell + 1)$ and $w^2(N(T, P)) = w^2(\{a_0, a_1, \ldots, a_{\ell+1}\}) \leq 9\ell + 8$. We conclude that $\ell \geq 37$ as otherwise T would be an improving subset of vertices. Therefore, the length of the above shortest path is $d(a, b) \geq 76$ and it contains at least $\ell \geq 37$ vertices of P_3.

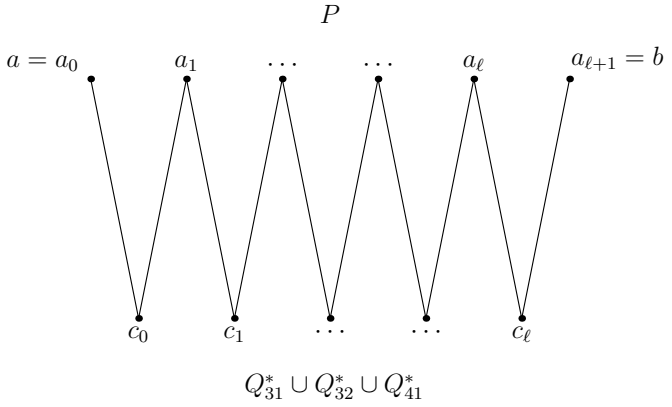

FIGURE 5.3: The configuration of the shortest path connecting $a, b \in P_4 \cup P_5$.

To prove the second half of the lemma, we notice that graph H' is connected. For every vertex $a \in P_4 \cup P_5$, we pick arbitrarily another vertex $b \in P_4 \cup P_5$ and consider a shortest path connecting them in graph H' that does not contain any other vertex from $P_4 \cup P_5$:

$$\langle a = a_0, c_0, a_1, c_1, \ldots, a_\ell, c_\ell, a_{\ell+1} = b \rangle,$$

for some $\ell \geq 37$. Initially every vertex of $P_4 \cup P_5$ is worth 1 token; through this path, vertex a distributes its 1 token evenly to vertices $a_1, a_2, \ldots, a_{18}$, which are all vertices of P_3. After every vertex of $P_4 \cup P_5$ has distributed its token, from $\ell \geq 37$ we conclude that every vertex of P_3 receives no more than $\frac{1}{18}$ token. Therefore,

$$p_4 + p_5 \leq \frac{1}{18} p_3.$$

This proves the lemma. □

From Lemma 5.10, we see that the number of 4-sets and 5-sets in the computed set packing P is very small compared against the number of 3-sets. In the following Lemma 5.11

we prove that $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$ through an amortized analysis. By Eq. (5.4), Lemma 5.11 is sufficient to prove Lemma 5.4.

Lemma 5.11. $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$.

PROOF. The proof of the lemma is through an amortized analysis, and is done via five distinct cases. In the following five lemmas (Lemmas 5.12, 5.13, 5.14, 5.15, 5.18), we assign 2 tokens for each vertex of Q^*_{31} and 1 token for each vertex of $Q^*_{32} \cup Q^*_{41}$. So we have a total of $2q^*_{31} + q^*_{32} + q^*_{41}$ tokens. We will prove that $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$ by distributing these tokens to the vertices of P.

We consider the following five distinct cases of graph H', which are separately dealt with in Lemmas 5.12, 5.13, 5.14, 5.15 and 5.18:

Case 1. $q^*_{31} = q^*_{41} = 0$,

Case 2. $q^*_{31} = 1$ and $q^*_{41} = 0$,

Case 3. $q^*_{31} = 0$ and $q^*_{41} = 1$,

Case 4. $q^*_{31} = 0$ and $q^*_{41} = 2$ with either $q^{*1}_{41} = 2$ or $q^{*2}_{41} = 2$,

Case 5. $q^*_{31} + q^*_{41} \geq 2$ excluding Case 4.

(Proof of Lemma 5.11 to be continued)

Lemma 5.12. (Case 1) *If* $q^*_{31} = q^*_{41} = 0$, *then* $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$.

Proof. From Lemma 5.8(c), every set of Q^*_{32} must intersect a set of P_3. If $p_4 + p_5 \leq 1$, then we have $2q^*_{32} \leq 3p_3 + 5$. It follows that when $p_3 \geq 5$, $2q^*_{32} \leq 3p_3 + 5 \leq 4p_3$ and thus $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$. When $p_3 = 4$ $(3, 2, 1, 0$, respectively), $w^2(P) \leq 40$ $(31, 22, 13, 4$, respectively) and thus $q^*_{32} \leq 4$ $(3, 2, 1, 0$, respectively) by algorithm GREEDY; that is, $q^*_{32} \leq p_3$, and consequently $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$.

If $p_4 + p_5 > 1$, every set of Q^*_{32} distributes $\frac{1}{2}$ token to each adjacent set of P. Note that every i-set of P receives at most $\frac{i}{2}$ token, by Lemma 5.10,

$$q^*_{32} \leq \sum_{i=3}^{5} \frac{i}{2} p_i \leq \frac{3}{2} p_3 + \frac{5}{2}(p_4 + p_5) \leq \frac{3}{2} p_3 + \frac{5}{2} \times \frac{1}{18} p_3 = \frac{59}{36} p_3,$$

and consequently $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$. □

Lemma 5.13. (Case 2) *If* $q^*_{31} = 1$, $q^*_{41} = 0$, *then* $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$.

Proof. From Lemma 5.8(a, c), the unique set of Q^*_{31} must intersect a set of P_3 and every set of Q^*_{32} must intersect a set of P_3. If $p_4 + p_5 \leq 1$, then we have $1 + 2q^*_{32} \leq 3p_3 + 5$, or $2q^*_{31} + q^*_{32} \leq \frac{3}{2}p_3 + 4$. It follows that when $p_3 \geq 8$, $2q^*_{31} + q^*_{32} \leq 2p_3$ and thus $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$; when $2 \leq p_3 \leq 7$, $9(q^*_{31} + q^*_{32}) \leq w^2(P) \leq 9p_3 + 4$ by algorithm GREEDY, and consequently $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$; when $p_3 = 1$, $w^2(P) \leq 13$ and thus $q^*_{32} = 0$ by algorithm GREEDY, and consequently $2q^*_{31} + q^*_{32} + q^*_{41} = 2 \leq 2p_3 + p_4$.

If $p_4 + p_5 > 1$, the unique set of Q^*_{31} distributes its 2 tokens to the adjacent set of P_3, and every set of Q^*_{32} distributes $\frac{1}{2}$ token to each adjacent set of P. Note that every i-set of P receives at most $\frac{i}{2}$ token, except the one adjacent to the unique set of Q^*_{31}, by Lemma 5.10 we have,

$$2q^*_{31} + q^*_{32} \leq \sum_{i=3}^{5} \frac{i}{2}p_i + \frac{3}{2} \leq \frac{3}{2}p_3 + \frac{5}{2}(p_4 + p_5) + \frac{3}{2} \leq \frac{59}{36}p_3 + \frac{3}{2} < 2p_3,$$

and consequently $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$. □

Lemma 5.14. (Case 3) *If* $q^*_{31} = 0$, $q^*_{41} = 1$, *then* $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$.

Proof. From Lemma 5.8(b, c), the unique set of Q^*_{41} must intersect a set of $P_3 \cup P_4$ and every set of Q^*_{32} must intersect a set of P_3. If $p_4 + p_5 \leq 1$, then we have $2q^*_{32} + 1 \leq 3p_3 + 5$, or $q^*_{32} + q^*_{41} \leq \frac{3}{2}p_3 + 3$. It follows that when $p_3 \geq 6$, $q^*_{32} + q^*_{41} \leq 2p_3$ and thus $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$; when $1 \leq p_3 \leq 5$, $9q^*_{32} + 4 \leq w^2(P) \leq 9p_3 + 4$ by algorithm GREEDY, and consequently $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$; when $p_3 = 0$, $p_4 = 1$ and $w^2(P) = 4$ and thus $q^*_{32} = 0$ by algorithm GREEDY, and consequently $2q^*_{31} + q^*_{32} + q^*_{41} = 1 \leq 2p_3 + p_4$.

If $p_4 + p_5 > 1$, the unique set of Q^*_{41} distributes its 1 token to the adjacent set of $P_3 \cup P_4$, and every set of Q^*_{32} distributes $\frac{1}{2}$ token to each adjacent set of P. Note that every i-set of P receives at most $\frac{i}{2}$ token, except the one adjacent to the unique set of Q^*_{41}, by Lemma 5.10 we have,

$$q^*_{32} + q^*_{41} \leq \sum_{i=3}^{5} \frac{i}{2}p_i + \frac{1}{2} \leq \frac{59}{36}p_3 + \frac{1}{2} < 2p_3,$$

and consequently $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$. □

Lemma 5.15. (Case 4) *If* $q^*_{31} = 0$ *and* $q^*_{41} = 2$ *with either* $q^{*1}_{41} = 2$ *or* $q^{*2}_{41} = 2$, *then* $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$.

Proof. From Lemma 5.8(b1, b2, c), each of these two sets of Q^*_{41} must intersect a set of P_3 and every set of Q^*_{32} must intersect a set of P_3. If $p_4 + p_5 \leq 1$, then we have

$2q^*_{32} + 2 \leq 3p_3 + 5$, or $q^*_{32} + q^*_{41} \leq \frac{3}{2}p_3 + \frac{7}{2}$. It follows that when $p_3 \geq 7$, $q^*_{32} + q^*_{41} \leq 2p_3$ and thus $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$; when $2 \leq p_3 \leq 6$, $9q^*_{32} + 8 \leq w^2(P) \leq 9p_3 + 4$ by algorithm GREEDY, and consequently $2q^*_{31} + q^*_{32} + q^*_{41} \leq p_3 + \frac{14}{9} \leq 2p_3 + p_4$; when $p_3 = 1$, $q^*_{32} = 0$ by algorithm GREEDY, and consequently $2q^*_{31} + q^*_{32} + q^*_{41} = 2 \leq 2p_3 + p_4$.

If $p_4 + p_5 > 1$, each of the two sets of Q^*_{41} distributes its 1 token to the adjacent set of P_3, and every set of Q^*_{32} distributes $\frac{1}{2}$ token to each adjacent set of P. Note that every i-set of P receives at most $\frac{i}{2}$ token, except the one(s) adjacent to the two sets of Q^*_{41}, by Lemma 5.10 we have,

$$q^*_{32} + q^*_{41} \leq \sum_{i=3}^{5} \frac{i}{2}p_i + 1 \leq \frac{59}{36}p_3 + 1 < 2p_3,$$

and consequently $2q^*_{31} + q^*_{32} + q^*_{41} \leq 2p_3 + p_4$. □

Note that in Case 5 we have $q^*_{31} + q^*_{41} \geq 2$, but if $q^*_{31} = 0$ and $q^*_{41} = 2$ then we should have $q^{*1}_{41} \neq 2$ and $q^{*2}_{41} \neq 2$. Note that we do not have the scenario where $q^*_{41} = 2$ but $q^{*1}_{41} = 1$, because sets of Q^{*1}_{41} always come in pairs by definition. That is, if $q^*_{31} = 0$ and $q^*_{41} = 2$ then we should have $q^{*3}_{41} \geq 1$.

Lemma 5.16. *In* Case 5, $q^*_{32} \geq 35$ *and for any set* $a \in Q^*_{31} \cup Q^*_{41}$ *in graph* H',

(a) *if* $a \in Q^*_{31} \cup Q^{*3}_{41}$, *then for any other set* $b \in Q^*_{31} \cup Q^*_{41}$ *we have distance* $d(a, b) \geq 74$;

(b) *if* $a \in Q^{*1}_{41}$, *then there is exactly one other set* $b \in Q^*_{41}$ *such that distance* $d(a, b) < 74$; *furthermore,* $b \in Q^{*1}_{41}$ *too and they come as a pair defining their memberships of* Q^{*1}_{41};

(c) *if* $a \in Q^{*2}_{41}$, *then there is at most one other set* $b \in Q^*_{41}$ *such that distance* $d(a, b) < 38$; *furthermore, if such a set* b *exists, then* $b \in Q^{*2}_{41}$ *too.*

Proof. The proof is similar to the proof of Lemma 5.10. We first notice that the scenario where there are only two sets in $Q^*_{31} \cup Q^*_{41}$ and they are adjacent to the same set of P is included in Case 4. That is, in Case 5, we always have (at least) two sets of $Q^*_{31} \cup Q^*_{41}$ not adjacent to the same set of P. Let a and b denote these two sets of $Q^*_{31} \cup Q^*_{41}$.

Since H' is bipartite, the shortest path connecting a and b has an even length and is denoted as

$$\langle a = a_0, c_0, a_1, c_1, \ldots, a_\ell, c_\ell, a_{\ell+1} = b \rangle,$$

for some $\ell \geq 1$. Since every vertex a_i $(i = 1, 2, \ldots, \ell)$ on the path has degree at least 2, it has to belong to Q^*_{32} and consequently it has degree exactly 2 in graph H' (see FIGURE 5.4).

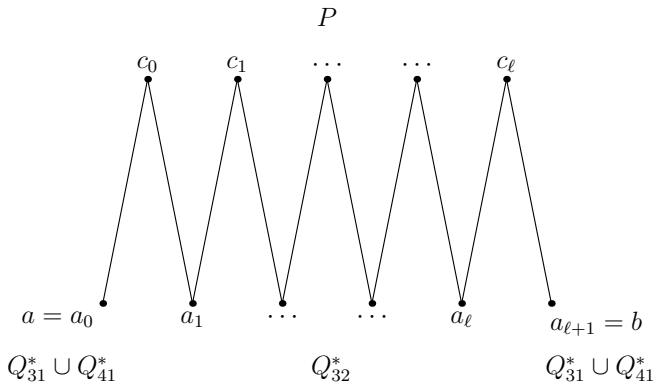

FIGURE 5.4: The configuration of the shortest path connecting $a, b \in Q^*_{31} \cup Q^*_{41}$.

Let $T = \{a_0, a_1, \ldots, a_{\ell+1}\}$. Clearly, $N(T, P) = \{c_0, c_1, \ldots, c_\ell\}$. If $a \in Q^*_{31}$, we have $w^2(T) \geq 9(\ell + 1) + 4$ and $w^2(N(T, P)) \leq 9(\ell + 1)$. The termination condition of algorithm GREEDY implies that the size $|T| = \ell + 2 > 37$, and thus $d(a, b) \geq 74$. One part of Item (a) is proved.

Consider the remaining case where both a and b are in Q^*_{41}. If at least one of $N(T, P) = \{c_0, c_1, \ldots, c_\ell\}$ is in $P_4 \cup P_5$, then we have $w^2(T) = 9\ell + 8$ and $w^2(N(T, P)) \leq 9\ell + 4$. Again the termination condition of algorithm GREEDY implies that $|T| = \ell + 2 > 37$, and thus $d(a, b) \geq 74$. The other part of Item (a) is proved.

If none of $N(T, P) = \{c_0, c_1, \ldots, c_\ell\}$ is in $P_4 \cup P_5$, that is, all of them are in P_3, then we have $w^2(T) = 9\ell + 8$ and $w^2(N(T, P)) = 9\ell + 9$. Consider first $a \in Q^{*1}_{41}$, and let a' denote the other set of Q^{*1}_{41} which comes together with a as a pair (see Lemma 5.8(b1)), and let $T' = T \cup \{a'\}$. One clearly sees that $N(T', P) = N(T, P)$, $w^2(T') = 9\ell + 12 > w^2(N(T', P)) = 9\ell+9$. Therefore, again the termination condition of algorithm GREEDY implies that $|T| = \ell+2 > 37$, and thus $d(a, b) \geq 74$. This proves Item (b), as $d(a, a') = 2$.

We next consider both a and b are in Q^{*2}_{41}. Note that if $|Q^{*2}_{41}| = 2$, that is, Q^{*2}_{41} contains only two sets a and b, then Item (c) is proved. We therefore let b' denote any set of Q^{*2}_{41} other than a and b. Using the same argument as in the last paragraph, if b' is adjacent to any of $N(T, P) = \{c_0, c_1, \ldots, c_\ell\}$, then $|T| = \ell+2 > 37$, and thus $d(a, b) \geq 74$; otherwise we let the shortest path connecting a and b' be

$$\langle a = a'_0, c'_0, a'_1, c'_1, \ldots, a'_{\ell'}, c'_{\ell'}, a'_{\ell'+1} = b' \rangle$$

for some $\ell' \geq 1$, which has the maximum overlap with the shortest path connecting a and b. This maximum overlap means the induced subgraph by these two paths does not contain a cycle. Let $T' = \{a'_0, a'_1, \ldots, a'_{\ell'+1}\}$ and $N(T', P) = \{c'_0, c'_1, \ldots, c'_{\ell'}\} \subset P_3$, and we have $w^2(T') = 9\ell' + 8$ and $w^2(N(T', P)) = 9\ell' + 9$. Let the overlapping sub-path be $\langle a = a_0, c_0, a_1, c_1, \ldots, a_o, c_o \rangle$ (see FIGURE 5.5).

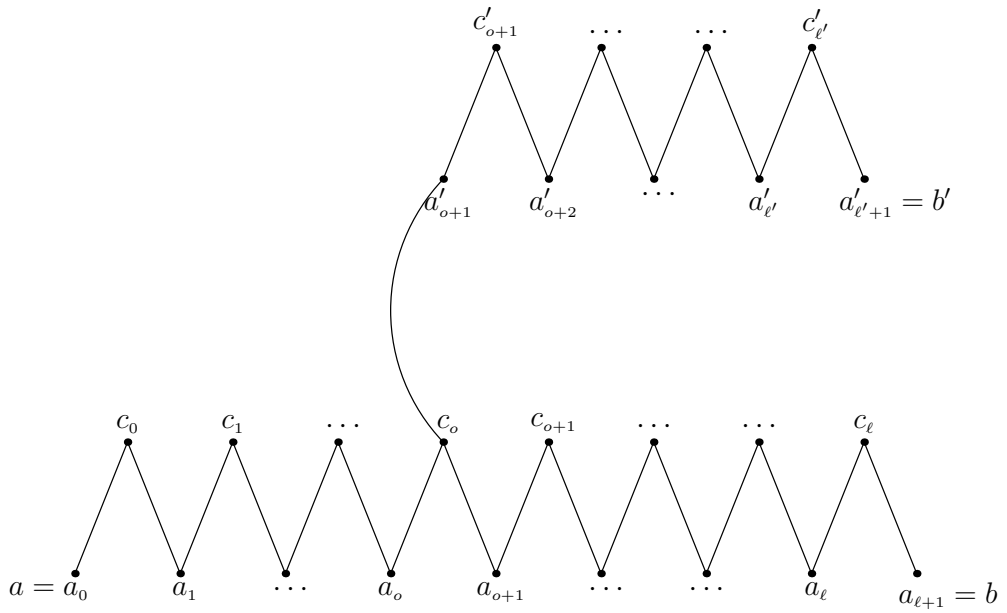

FIGURE 5.5: The configuration of the overlapping shortest paths connecting $a, b \in Q^{*2}_{41}$ and connecting $a, b' \in Q^{*2}_{41}$.

We then have

$$T \cup T' = \{a_0, a_1, \ldots, a_o, a_{o+1}, \ldots, a_{\ell+1}, a'_{o+1}, \ldots, a'_{\ell'+1}\},$$
$$N(T \cup T', P) = \{c_0, c_1, \ldots, c_o, c_{o+1}, \ldots, c_\ell, c'_{o+1}, \ldots, c'_{\ell'}\},$$

and thus

$$w^2(T \cup T') = 9(\ell + \ell' - o) + 12,$$
$$w^2(N(T \cup T', P)) = 9(\ell + \ell' - o) + 9.$$

Therefore, again the termination condition of algorithm GREEDY implies that $|T \cup T'| = \ell + \ell' - o + 3 > 37$, or $\ell + \ell' - o > 34$. Thus $\max\{\ell, \ell'\} \geq 18$, implying that $\max\{d(a, b), d(a, b')\} \geq 38$. This proves Item (c) of the lemma.

Lastly, to estimate the quantity q^*_{32}, we know from all the above proof that, either there is a pair of sets of $Q^*_{31} \cup Q^*_{41}$ at distance ≥ 74, and thus $q^*_{32} \geq 36$, or otherwise $q^{*2}_{41} \geq 3$ and from the last paragraph we have $q^*_{32} \geq \ell + \ell' - o \geq 35$. This completes the proof. □

Lemma 5.17. *In* Case 5, $q^*_{31} + \frac{1}{2}q^{*1}_{41} + \frac{1}{2}q^{*2}_{41} + q^{*3}_{41} \leq \frac{1}{9}q^*_{32}$.

Proof. The proof is similar to the proof of Lemma 5.10.

For every vertex $a \in Q^*_{31} \cup Q^{*3}_{41}$, by Lemma 5.16(a) we pick arbitrarily another vertex $b \in Q^*_{31} \cup Q^*_{41}$ and consider a shortest path connecting them in graph H' that does not

contain any other vertex from $Q_{31}^* \cup Q_{41}^*$:

$$\langle a = a_0, c_0, a_1, c_1, \ldots, a_\ell, c_\ell, a_{\ell+1} = b \rangle,$$

for some $\ell \geq 36$.

For any two vertices $a, a' \in Q_{41}^{*1}$ that come as a pair to define their memberships, by Lemma 5.16(b) use exactly one of them say a (and mark a') and pick arbitrarily another vertex $b \in Q_{31}^* \cup Q_{41}^*$ ($b \neq a'$) to consider their shortest path in graph H' that does not contain any other vertex from $Q_{31}^* \cup Q_{41}^*$:

$$\langle a = a_0, c_0, a_1, c_1, \ldots, a_\ell, c_\ell, a_{\ell+1} = b \rangle,$$

for some $\ell \geq 36$.

For any vertex $a \in Q_{41}^{*2}$, if there is another vertex $a' \in Q_{41}^{*2}$ such that $d(a, a') < 38$, by Lemma 5.16(c) use the one of them say a (and mark a') for which there is another vertex $b \in Q_{31}^* \cup Q_{41}^*$ such that $d(a, b) \geq 38$ and consider the shortest path connecting a and b that does not contain any other vertex from $Q_{31}^* \cup Q_{41}^*$:

$$\langle a = a_0, c_0, a_1, c_1, \ldots, a_\ell, c_\ell, a_{\ell+1} = b \rangle,$$

for some $\ell \geq 18$.

Initially every vertex of $Q_{31}^* \cup Q_{41}^*$ is worth 1 token, except those marked vertices; through the picked path, vertex a distributes its 1 token evenly to vertices $a_1, a_2, \ldots, a_9$, which are all vertices of Q_{32}^*. After every vertex of $Q_{31}^* \cup Q_{41}^*$ has distributed its token, from $\ell \geq 18$ we conclude that every vertex of Q_{32}^* receives no more than $\frac{1}{9}$ token. Therefore,

$$q_{31}^* + \frac{1}{2} q_{41}^{*1} + \frac{1}{2} q_{41}^{*2} + q_{41}^{*3} \leq \frac{1}{9} q_{32}^*.$$

This proves the lemma. □

Lemma 5.18. (Case 5) *If $q_{31}^* \geq 1$ and $q_{41}^* \geq 1$, or if $q_{31}^* = 0$ and $q_{41}^* \geq 3$, or if $q_{31}^* = 0$ and $q_{41}^* = 2$ with $q_{41}^{*3} \geq 1$, then $2q_{31}^* + q_{32}^* + q_{41}^* \leq 2p_3 + p_4$.*

Proof. Recall that we assign 2 tokens for each vertex of Q_{31}^* and 1 token for each vertex of $Q_{32}^* \cup Q_{41}^*$. So we have a total of $2q_{31}^* + q_{32}^* + q_{41}^*$ tokens. These tokens are distributed to the vertices of P in the same way as before, that each vertex of Q_{31}^* distributes its 2 tokens to the adjacent vertex of P, each vertex of Q_{41}^* distributes its 1 token to the adjacent vertex of P, and each vertex of Q_{32}^* distributes its $\frac{1}{2}$ token to every adjacent

vertex of P. Let $t(a)$ denote the total tokens received by a vertex $a \in P$; we have

$$t(a) \leq \begin{cases} 3, & a \in P_3^1; \\ \frac{5}{2}, & a \in P_3^2; \\ 2, & a \in P_3^3; \\ \frac{3}{2}, & a \in P_3^4; \\ \frac{5}{2}, & a \in P_4; \\ \frac{5}{2}, & a \in P_5. \end{cases} \tag{5.8}$$

It follows that

$$\begin{aligned} & 2q_{31}^* + q_{32}^* + q_{41}^* \\ \leq & \left(3p_3^1 + \frac{5}{2}p_3^2 + 2p_3^3 + \frac{3}{2}p_3^4\right) + \frac{5}{2}p_4 + \frac{5}{2}p_5 \\ = & \left(3p_3^1 + \frac{5}{2}p_3^2 + 2p_3^3 + \frac{3}{2}\left(p_3 - p_3^1 - p_3^2 - p_3^3\right)\right) + \frac{5}{2}p_4 + \frac{5}{2}p_5 \\ = & \left(\frac{3}{2}p_3 + \frac{3}{2}p_3^1 + p_3^2 + \frac{1}{2}p_3^3\right) + \frac{5}{2}p_4 + \frac{5}{2}p_5 \\ = & \frac{3}{2}p_3 + \left(\frac{3}{2}q_{31}^* + \frac{1}{2}q_{41}^{*1} + \frac{1}{2}q_{41}^{*2}\right) + \frac{5}{2}p_4 + \frac{5}{2}p_5 \text{ (by Lemma 5.9)} \\ = & \frac{3}{2}p_3 + \frac{3}{2}\left(q_{31}^* + \frac{1}{3}q_{41}^{*1} + \frac{1}{3}q_{41}^{*2}\right) + \frac{5}{2}p_4 + \frac{5}{2}p_5 \text{ (by manipulation)} \\ \leq & \frac{3}{2}p_3 + \frac{3}{2} \times \frac{1}{9}q_{32}^* + \frac{5}{2}p_4 + \frac{5}{2}p_5 \text{ (by Lemma 5.17)} \\ \leq & \frac{6}{5}\left(\frac{3}{2}p_3 + \frac{5}{2}p_4 + \frac{5}{2}p_5\right) \text{ (by manipulation)} \\ = & \frac{9}{5}p_3 + 3\left(p_4 + p_5\right). \end{aligned} \tag{5.9}$$

If $p_4 + p_5 \leq 1$, Eq.(5.9) becomes $2q_{31}^* + q_{32}^* + q_{41}^* \leq \frac{9}{5}p_3 + 3$. From Lemma 5.16 we know that there is a shortest path of length at least 38 in graph H' connecting two vertices of $Q_{31}^* \cup Q_{41}^*$. Therefore, $p_3 + p_4 + p_5 \geq 19$. It follows that $p_3 \geq 18$, and thus $2q_{31}^* + q_{32}^* + q_{41}^* < 2p_3 \leq 2p_3 + p_4$.

If $p_4 + p_5 > 1$, by Lemma 5.10, Eq. (5.9) becomes

$$2q_{31}^* + q_{32}^* + q_{41}^* \leq \frac{9}{5}p_3 + 3 \times \frac{1}{18}p_3 \leq 2p_3 \leq 2p_3 + p_4.$$

This proves the lemma. □

Proof (of Lemma 5.11, continued). The five distinct cases of graph H', separately dealt with in Lemmas 5.12, 5.13, 5.14, 5.15 and 5.18, are complete by considering all possible

values of quantity $q_{31}^* + q_{41}^*$. And in each case we have proved that $2q_{31}^* + q_{32}^* + q_{41}^* \leq 2p_3 + p_4$. □

5.4 A $0.6k$-approximation algorithm for k-MCIP

Given an instance of the k-MCIP problem $\{X_1, X_2, \ldots, X_k\}$, we first divide these k multisets into $\lfloor k/2 \rfloor$ pairs $\{X_{2i-1}, X_{2i}\}$, $i = 1, 2, \ldots, \lfloor k/2 \rfloor$, plus the last multiset X_k if k is odd. Next, we run algorithm Apx65 on each pair $\{X_{2i-1}, X_{2i}\}$ to obtain a solution $Z_i = \mathrm{CIP}_{\mathrm{Apx65}}(X_{2i-1}, X_{2i})$, for $i = 1, 2, \ldots, \lfloor k/2 \rfloor$, plus $Z_{(k+1)/2} = X_k$ if k is odd. We continue this dividing and running Apx65 on $\{Z_1, Z_2, \ldots, Z_{\lceil (k+1)/2 \rceil}\}$ if $\lceil (k+1)/2 \rceil \geq 2$, and repeat until we have only one multiset left, denoted as $\mathrm{CIP}_{\mathrm{final}}$. Clearly, $\mathrm{CIP}_{\mathrm{final}}$ is a common integer partition of the given multisets $X_1, X_2, \ldots, X_k$.

Theorem 5.19. k-MCIP *admits a* $0.6k$*-approximation algorithm when* k *is even, or a* $(0.6k + 0.4)$*-approximation algorithm when* k *is odd.*

Proof. The algorithm in the last paragraph producing a feasible solution $\mathrm{CIP}_{\mathrm{final}}$ runs in polynomial time. We next estimate its performance, and assume that k is even. By Theorem 5.7, we have $|Z_i| < \frac{6}{5}|\mathrm{OPT}(X_{2i-1}, X_{2i})|$, for $i = 1, 2, \ldots, k/2$. Let OPT denote the minimum common integer partition for $X_1, X_2, \ldots, X_k$. One clearly sees that $|\mathrm{OPT}(X_{2i-1}, X_{2i})| \leq |\mathrm{OPT}|$, and from Lemma 5.1 we have

$$|\mathrm{CIP}_{\mathrm{final}}| < \sum_{i=1}^{k/2} |Z_i| < \sum_{i=1}^{k/2} \frac{6}{5}|\mathrm{OPT}| = \frac{3k}{5}|\mathrm{OPT}|.$$

When k is odd,

$$|\mathrm{CIP}_{\mathrm{final}}| < \sum_{i=1}^{(k-1)/2} |Z_i| + |X_k| < \sum_{i=1}^{(k-1)/2} \frac{6}{5}|\mathrm{OPT}| + |\mathrm{OPT}| = \frac{3k+2}{5}|\mathrm{OPT}|,$$

using $|X_k| \leq |\mathrm{OPT}|$ from Lemma 5.1. This completes the proof. □

5.5 Conclusions and future work

We studied the minimum common integer partition problem (k-MCIP) and improved the previous best approximation ratio for the 2-MCIP. The main idea is that we applied a novel local search method inspired by the local structure of the 2-MCIP and the similarity between the 2-MCIP and the weighted t-set packing problem. There are many directions for the future work for the k-MCIP. One way is to find more specific local

structure of the 2-MCIP. The second way is to look deep into the relation of the 2-MCIP and the unweighted t-set packing problem or even the hypergraph matching problem and then to design new local search strategy. The third way is to investigate the k-MCIP with larger $k > 2$ to see whether our current method is still available. Another way is to design efficient heuristics for the real application because the k-MCIP has some biological meanings as we introduced in the first section. Here we need to mention that our approximation algorithm for the 2-MCIP has been modified to an online toy game. [2] Amazingly, our algorithm is quite fast and returns the almost optimal solution in most cases. So it would be interesting to investigate our algorithm under the smoothed analysis model to see its "real" theoretical performance.

[2]This game was implemented in summer 2014 by Yifan Song, a summer student from the University of Waterloo. Here is the link for the game: http://beiseker.cs.ualberta.ca:8080/MCIP/.

Chapter 6

Minimum Independent Dominating Set (MIDS) Problem[1]

6.1 Introduction

An *independent set* in a graph $G = (V, E)$ is a subset of vertices that are pair-wise non-adjacent to each other. The independence number of G, denoted by $\alpha(G)$, is the size of a maximum independent set in G. One close notion to independent set is the *dominating set*, which refers to a subset of vertices such that every vertex of the graph is either in the subset or is adjacent to some vertex in the subset. In fact, an independent set becomes a dominating set if and only if it is maximal. The size of a minimum independent dominating set of G is denoted by $i(G)$, while the domination number of G, or the size of a minimum dominating set of G, is denoted by $\gamma(G)$. It follows that $\gamma(G) \leq i(G) \leq \alpha(G)$.

Another related notion is the (vertex) *coloring* of G, in which two adjacent vertices must be colored differently. Note that any subset of vertices colored the same in a coloring of G is necessarily an independent set. The *chromatic number* $\chi(G)$ of G is the minimum number of colors in a coloring of G. Clearly, $\alpha(G) \cdot \chi(G) \geq |V|$.

The independence number $\alpha(G)$ and the domination number $\gamma(G)$ (and the chromatic number $\chi(G)$) have received numerous studies due to their central roles in graph theory and theoretical computer science. Their exact values are NP-hard to compute [44], and hard to approximate. Raz and Safra showed that the domination number cannot be approximated within $(1-\epsilon)\log|V|$ for any fixed $\epsilon > 0$, unless NP $\subset$ DTIME($|V|^{\log\log|V|}$) [39, 82]; Zuckerman showed that neither the independence number nor the chromatic number can be approximated within $|V|^{1-\epsilon}$ for any fixed $\epsilon > 0$, unless P = NP [114]; for $i(G)$, Halldórsson proved that it is also hard to approximate within $|V|^{1-\epsilon}$ for any fixed $\epsilon > 0$, unless NP $\subset$ DTIME($2^{o(|V|)}$) [47].

[1] This chapter is based on [100, 101].

The above inapproximability results are for the worst case. For analyzing the average case performance of approximation algorithms, a probability distribution of the input graphs must be assumed and the most widely used distribution of graphs on n vertices is the random graph $G(n,p)$, which is a graph on n vertices, and each edge is chosen to be an edge of G independently with a probability p, where $0 \leq p = p(n) \leq 1$. A graph property holds *asymptotically almost surely* (a.a.s.) in $G(n,p)$ if the probability that a graph drawn according to the distribution $G(n,p)$ has the property tends to 1 as n tends to infinity [16].

Let $\mathbb{L}n = \log_{1/(1-p)} n$. Bollobás [15] and Łuczak [65] showed that a.a.s. $\chi(G(n,p)) = (1+o(1))n/\mathbb{L}n$ for a constant p and $\chi(G(n,p)) = (1+o(1))np/(2\ln(np))$ for $c/n \leq p(n) \leq o(1)$ where c is a constant. It follows from these results that a.a.s. $\alpha(G(n,p)) = (1-o(1))\mathbb{L}n$ for a constant p and $\alpha(G(n,p)) = (1-o(1))2\ln(np)/p$ for $C/n \leq p \leq o(1)$. The greedy algorithm, which colors vertices of $G(n,p)$ one by one and picks each time the first available color for a current vertex, is known to produce a.a.s. in $G(n,p)$ with $p \geq n^{\epsilon-1}$ a coloring whose number of colors is larger than the $\chi(G(n,p))$ by only a constant factor (see Chapter 11 of the monograph of Bollobás [16]). Hence the largest color class produced by the greedy algorithm is a.a.s. smaller than $\alpha(G(n,p))$ only by a constant factor.

For the domination number $\gamma(G(n,p))$, Wieland and Godbole showed that a.a.s. it is equal to either $\lfloor \mathbb{L}n - \mathbb{L}((\mathbb{L}n)(\ln n)) \rfloor + 1$ or $\lfloor \mathbb{L}n - \mathbb{L}((\mathbb{L}n)(\ln n)) \rfloor + 2$, for a constant p or a suitable function $p = p(n)$ [110]. It follows that a.a.s. $i(G(n,p)) \geq \lfloor \mathbb{L}n - \mathbb{L}((\mathbb{L}n)(\ln n)) \rfloor + 1$. Recently, Wang proved for $i(G(n,p))$ an a.a.s. upper bound of $\lfloor \mathbb{L}n - \mathbb{L}((\mathbb{L}n)(\ln n)) \rfloor + k + 1$, where $k = \max\{1, \mathbb{L}2\}$ [108].

Average case performance analysis of an approximation algorithm over random instances could be inconclusive, because the random instances usually have very special properties that distinguish them from real-world instances. For instance, for a constant p, the random graph $G(n,p)$ is expected to be dense. On the other hand, an approximation algorithm performs very well on most random instances can fail miserably on some "hard" instances. For instance, it has been shown by Kučera [62] that for any fixed $\epsilon > 0$ there exists a graph G on n vertices for which, even after a random permutation of vertices, the greedy algorithm produces a.a.s. a coloring using at least $n/\log_2 n$ colors, while $\chi(G) \leq n^{\epsilon}$.

We study the approximability of the minimum independent dominating set (MIDS) problem under the smoothed analysis, and we present a simple deterministic greedy algorithm beating the strong inapproximability bound of $n^{1-\epsilon}$, with polynomial expected running time. The MIDS problem, and the closely related independent set and dominating set problems, have important applications in wireless networks, and have been

studied extensively in the literature. Our probabilistic model is the smoothed extension of random graph $G(n,p)$ (also called semi-random graphs in [67]), proposed by Spielman and Teng [92]: given a graph $G=(V,E)$, we define its perturbed graph $\mathfrak{g}(G,p)$ by negating the existence of edges independently with a probability of $p>0$. That is, $\mathfrak{g}(G,p)$ has the same vertex set V as G but it contains edge e with probability p_e, where $p_e=1-p$ if $e\in E$ or otherwise $p_e=p$. For sufficiently large p, Manthey and Plociennik presented an algorithm approximating the independence number $\alpha(\mathfrak{g}(G,p))$ with a worst-case performance ratio $O(\sqrt{np})$ and with polynomial expected running time [67].

Re-define $\mathbb{L}n=\log_{1/p} n$. We first prove on $\gamma(\mathfrak{g}(G,p))$, and thus on $i(\mathfrak{g}(G,p))$ as well, an a.a.s. lower bound of $\mathbb{L}n-\mathbb{L}((\mathbb{L}n)(\ln n))$ if $p>\frac{1}{n}$. We then prove on $\alpha(\mathfrak{g}(G,p))$, and thus on $i(\mathfrak{g}(G,p))$ as well, an a.a.s. upper bound of $2\ln n/p$ if $p<\frac{1}{2}$ or $2\ln n/(1-p)$ otherwise. Given the a.a.s. values of $\alpha(G(n,p))$ and $i(G(n,p))$ in random graph $G(n,p)$, our upper bound comes with no big surprise; nevertheless, our upper bound is derived by a direct counting process which might be interesting by itself. Furthermore, we extend our counting techniques to prove on $i(\mathfrak{g}(G,p))$ a tail bound that, when $4\ln^2 n/n < p\le\frac{1}{2}$, $\Pr[i(\mathfrak{g}(G,p))\ge\sqrt{4n/p}]\le 2^{-n}$. We then present a simple greedy algorithm to approximate $i(\mathfrak{g}(G,p))$, and prove that its worst case performance ratio is $\sqrt{4n/p}$ and its expected running time is polynomial.

6.2 A.a.s. bounds on the independent domination number

We need the following several facts.

Fact 6.1. $e^{\frac{x}{1+x}}\le 1+x\le e^x$ *holds for all* $x\in[-1,1]$.

Fact 6.2. $\left(\frac{n}{r}\right)^r\le\binom{n}{r}\le\left(\frac{ne}{r}\right)^r$ *holds for all* $r=0,1,2,\ldots,n$.

Fact 6.3. (Jensen's Inequality) *For a real convex function* $f(x)$, *numbers* x_1, x_2, ..., x_n *in its domain, and positive weights* a_i, $f\left(\frac{\sum a_i x_i}{\sum a_i}\right)\le\frac{\sum a_i f(x_i)}{\sum a_i}$; *the inequality is reversed if* $f(x)$ *is concave.*

Given any graph $G=(V,E)$, let $\mathfrak{g}(G,p)$ denote its perturbed graph, which has the same vertex set V as G and contains edge e with a probability of

$$p_e=\begin{cases}1-p, & \text{if } e\in E,\\ p, & \text{otherwise.}\end{cases}$$

6.2.1 An a.a.s. lower bound

Recall that $\gamma(\mathfrak{g}(G,p))$ and $i(\mathfrak{g}(G,p))$ are the domination number and the independent domination number of $\mathfrak{g}(G,p)$, respectively. Also, $\mathbb{L}n = \log_{1/p} n$.

Theorem 6.4. *For any graph $G=(V,E)$ of large enough size, say $n>30$, and $\frac{2\ln n}{n} < p < 1-\frac{2\ln n}{n}$, a.a.s.*

$$\gamma(\mathfrak{g}(G,p)) \geq \mathbb{L}n - \mathbb{L}((\mathbb{L}n)(\ln n)).$$

Proof. Let $\mathcal{S}_r$ be the collection of all r-subsets of vertices in $\mathfrak{g}(G,p)$, and these $\binom{n}{r}$ sets of $\mathcal{S}_r$ are ordered in some way. Define I_j^r as a boolean variable to indicate whether or not the j-th r-subset of $\mathcal{S}_r$, V_j, is a dominating set; set $X_r = \sum_j I_j^r$.

Clearly, $\gamma(\mathfrak{g}(G,p)) < r$ implies that there are size-r dominating sets. Therefore,

$$\Pr[\gamma(\mathfrak{g}(G,p)) < r] \leq \Pr[X_r \geq 1] \leq E(X_r),$$

where $E(X_r)$ is the expected value of X_r. (We abuse the notation E a little, but its meaning should be clear at every occurrence.)

For the j-th r-subset V_j, let E_j be the subset of induced edges on V_j from the original graph $G=(V,E)$; let $V_j^c = V - V_j$, the complement subset of vertices. Also, for each vertex $u \in V_j^c$, define $E(u,V_j) = \{(u,v) \in E \mid v \in V_j\}$, and its size $n_{uj} = |E(u,V_j)|$. Using Fact 6.1, we can estimate $E(X_r)$ as follows:

$$\begin{aligned}
E(X_r) = \sum_{j=1}^{\binom{n}{r}} E(I_j^r) &= \sum_{j=1}^{\binom{n}{r}} \prod_{u\in V_j^c} \left(1 - \prod_{v\in V_j} \left(1-p_{(u,v)}\right)\right) \\
&\leq \sum_{j=1}^{\binom{n}{r}} \prod_{u\in V_j^c} \exp\left(- \prod_{v\in V_j} \left(1-p_{(u,v)}\right)\right) \\
&= \sum_{j=1}^{\binom{n}{r}} \exp\left(- \sum_{u\in V_j^c} \prod_{v\in V_j} \left(1-p_{(u,v)}\right)\right) \\
&= \sum_{j=1}^{\binom{n}{r}} \exp\left(- \sum_{u\in V_j^c} p^{n_{uj}} (1-p)^{r-n_{uj}}\right) \\
&= \sum_{j=1}^{\binom{n}{r}} \exp\left(- \sum_{u\in V_j^c} \left(\frac{p}{1-p}\right)^{n_{uj}} (1-p)^r\right).
\end{aligned}$$

Since function $f(x) = (\frac{p}{1-p})^x$ is convex in the domain $[0, n]$, by Jensen's Inequality, the above becomes

$$E(X_r) \leq \sum_{j=1}^{\binom{n}{r}} \exp\left(-\left(\frac{p}{1-p}\right)^{\frac{1}{n-r}\sum_{u\in V_j^c} n_{uj}} (n-r)(1-p)^r \right).$$

Since function $g(x) = e^{-a^x b}$ with $a = (\frac{p}{1-p})^{\frac{1}{n-r}}$ and $b = (n-r)(1-p)^r$ is concave in the domain $[0, n^2]$, again by Jensen's Inequality, we further have

$$E(X_r) \leq \binom{n}{r} \exp\left(-\left(\frac{p}{1-p}\right)^{\frac{1}{(n-r)\binom{n}{r}}\sum_{j=1}^{\binom{n}{r}}\sum_{u\in V_j^c} n_{uj}} (n-r)(1-p)^r \right). \tag{6.1}$$

Recall that n_{uj} is number of edges in the original graph $G = (V, E)$ between u and vertices of V_j. Each edge $e \in E$ is thus counted towards the quantity $\left(\sum_{j=1}^{\binom{n}{r}}\sum_{u\in V_j^c} n_{uj}\right)$ exactly $2\binom{n-2}{r-1}$ times. That is,

$$\sum_{j=1}^{\binom{n}{r}}\sum_{u\in V_j^c} n_{uj} = 2\binom{n-2}{r-1}|E| = \frac{\binom{n}{r} r(n-r)|E|}{\binom{n}{2}}. \tag{6.2}$$

Using Eq. (6.2), Fact 6.2 and $r = \mathbb{L}n - \mathbb{L}((\mathbb{L}n)(\ln n))$, Eq. (6.1) becomes

$$\begin{aligned}
E(X_r) &\leq \binom{n}{r} \exp\left(-\left(\frac{p}{1-p}\right)^{\frac{r|E|}{\binom{n}{2}}} (n-r)(1-p)^r \right) \\
&\leq \binom{n}{r} \exp\left(-\left(\frac{p}{1-p}\right)^{r} (n-r)(1-p)^r \right) \\
&\leq \binom{ne}{r}^r \exp\Big(-p^r(n-r) \Big) \\
&\leq \exp\left(r\ln n + r - r\ln r - \frac{(\mathbb{L}n)(\ln n)}{n}(n-r) \right) \\
&= \exp((\mathbb{L}n)(\ln n) - \mathbb{L}((\mathbb{L}n)(\ln n))\ln n + r - r\ln r - (\mathbb{L}n)(\ln n) + r(\mathbb{L}n)(\ln n)/n) \\
&= \exp\left(-\mathbb{L}((\mathbb{L}n)(\ln n))\ln n - r\left(\ln r - (\mathbb{L}n)(\ln n)/n - 1\right)\right) \\
&\leq \exp\left(-\mathbb{L}((\mathbb{L}n)(\ln n))\ln n - r\left(\ln r - 2\right)\right). \qquad (6.3)
\end{aligned}$$

The right hand side in Eq. (6.3) approaches 0 when $n \to +\infty$. Since $\frac{2\ln n}{n} < p < 1 - \frac{2\ln n}{n}$

with large enough n, $n > 30$, guarantees $r \geq 1$, $\mathbb{L}n - \mathbb{L}((\mathbb{L}n)(\ln n))$ is an a.a.s. lower bound on $\gamma(\mathfrak{g}(G,p))$. This proves the theorem. □

Since $\Pr[i(\mathfrak{g}(G,p)) < r] \leq \Pr[\gamma(\mathfrak{g}(G,p)) < r]$, we have the following corollary:

Corollary 6.5. *For any graph $G = (V,E)$ of large enough size, say $n > 30$, and $\frac{2\ln n}{n} < p < 1 - \frac{2\ln n}{n}$, a.a.s.*

$$i(\mathfrak{g}(G,p)) \geq \mathbb{L}n - \mathbb{L}((\mathbb{L}n)(\ln n)).$$

6.2.2 An a.a.s. upper bound

Recall that $\alpha(\mathfrak{g}(G,p))$ is the independence number of $\mathfrak{g}(G,p)$.

Theorem 6.6. *For any graph $G = (V,E)$, a.a.s.*

$$\alpha(\mathfrak{g}(G,p)) \leq \begin{cases} \frac{2\ln n}{p}, & \text{if } p \in (\frac{2\ln n}{n}, \frac{1}{2}], \\ \frac{2\ln n}{1-p}, & \text{if } p \in [\frac{1}{2}, 1 - \frac{2\ln n}{n}). \end{cases}$$

Proof. Let $\mathcal{S}_r$ be the collection of all r-subsets of vertices in $\mathfrak{g}(G,p)$, and these $\binom{n}{r}$ sets of $\mathcal{S}_r$ are ordered in some way. Define I_j^r as a boolean variable to indicate whether or not the j-th r-subset of $\mathcal{S}_r$ is an independent set; set $X_r = \sum_j I_j^r$. Since $\alpha(\mathfrak{g}(G,p)) > r$ implies that there is at least one independent r-subset, i.e. $X_r > 0$, the probability of the event $\alpha(\mathfrak{g}(G,p)) > r$ is less than or equal to the probability of the event $X_r > 0$, i.e.

$$\Pr[\alpha(\mathfrak{g}(G,p)) > r] \leq \Pr[X_r > 0].$$

On the other hand, let A_j^r denote the event $I_j^r = 0$, i.e. the j-th r-subset is not independent. It follows that $X_r = 0$ is equivalent to the joint event $\cap_j A_j^r$, i.e.

$$\Pr[X_r = 0] = \Pr[\cap_j A_j^r] \geq \prod_j \Pr[A_j^r] = \prod_j (1 - \Pr[I_j^r = 1]).$$

Therefore, we have

$$\Pr[\alpha(\mathfrak{g}(G,p)) > r] \leq 1 - \prod_j (1 - \Pr[I_j^r = 1]). \tag{6.4}$$

Let E_j^r denote the subset of edges of $\mathfrak{g}(G,p)$, each of which connects two vertices in the j-th r-subset of $\mathcal{S}_r$. Note that $|E_j^r| \in [0, \binom{r}{2}]$. Among all the edges of E_j^r, assume there

are n_j^r of them coming from the original edge set E of G. It follows that

$$\Pr[I_j^r = 1] = \prod_{e \in E_j^r} (1 - p_e) = \left(\frac{p}{1-p}\right)^{n_j^r} (1-p)^{\binom{r}{2}}.$$

Using this and Fact 6.1 in Eq. (6.4) gives us

$$\begin{aligned}
\Pr[\alpha(\mathfrak{g}(G,p)) > r] &\leq 1 - \prod_j (1 - \Pr[I_j^r = 1]) \\
&\leq 1 - \prod_{j=1}^{\binom{n}{r}} \exp\left(-\frac{\Pr[I_j^r = 1]}{1 - \Pr[I_j^r = 1]}\right) \\
&= 1 - \exp\left(-\sum_{j=1}^{\binom{n}{r}} \frac{\Pr[I_j^r = 1]}{1 - \Pr[I_j^r = 1]}\right) \\
&= 1 - \exp\left(-\sum_{j=1}^{\binom{n}{r}} \frac{\left(\frac{p}{1-p}\right)^{n_j^r} (1-p)^{\binom{r}{2}}}{1 - \left(\frac{p}{1-p}\right)^{n_j^r} (1-p)^{\binom{r}{2}}}\right). \qquad (6.5)
\end{aligned}$$

Consider the function $f(x) = \frac{a^x b}{1 - a^x b}$ in Eq. (6.5), where $a = \frac{p}{1-p} > 0$, $b = (1-p)^{\binom{r}{2}} \in (0,1)$, and $0 \leq x \leq \binom{r}{2}$. Since its derivative

$$f'(x) = \frac{a^x b \ln a}{(1 - a^x b)^2} \begin{cases} < 0, & \text{if } a < 1, \\ = 0, & \text{if } a = 1, \\ > 0, & \text{if } a > 1, \end{cases}$$

$f(x)$ is strictly decreasing if $a < 1$, or strictly increasing if $a > 1$. Therefore, the maximum value of function $f(x)$ is achieved at $x = 0$ if $a \leq 1$, or at $x = \binom{r}{2}$ if $a \geq 1$.

When $p \leq \frac{1}{2}$, that is $a = \frac{p}{1-p} \leq 1$, Eq. (6.5) becomes

$$\begin{aligned}
\Pr[\alpha(\mathfrak{g}(G,p)) > r] &\leq 1 - \exp\left(-\sum_{j=1}^{\binom{n}{r}} \frac{(1-p)^{\binom{r}{2}}}{1 - (1-p)^{\binom{r}{2}}}\right) \\
&= 1 - \exp\left(-\binom{n}{r} \frac{(1-p)^{\binom{r}{2}}}{1 - (1-p)^{\binom{r}{2}}}\right). \qquad (6.6)
\end{aligned}$$

To prove $\Pr[\alpha(\mathfrak{g}(G,p)) > r] \to 0$ as $n \to +\infty$, we only need to prove that $\binom{n}{r}\frac{(1-p)^{\binom{r}{2}}}{1-(1-p)^{\binom{r}{2}}} \to 0$ as $n \to +\infty$. Using Fact 6.2, we have

$$\binom{n}{r}\frac{(1-p)^{\binom{r}{2}}}{1-(1-p)^{\binom{r}{2}}} = \frac{\binom{n}{r}}{\left(\frac{1}{1-p}\right)^{\binom{r}{2}}-1} \le \frac{\left(\frac{ne}{r}\right)^r}{\left(\frac{1}{1-p}\right)^{\binom{r}{2}}-1}. \tag{6.7}$$

Setting $r = 2\ln n/p$. We see that $r \to +\infty$ as $n \to +\infty$. On the other hand, when r is large enough, we have

$$\left(\frac{1}{1-p}\right)^{\binom{r}{2}} - 1 = \left(\frac{1}{1-p}\right)^{\binom{r}{2}}(1-o(1)). \tag{6.8}$$

Using Eq. (6.8) and Fact 6.1, when n is sufficiently large, Eq. (6.7) becomes

$$\begin{aligned}
\binom{n}{r}\frac{(1-p)^{\binom{r}{2}}}{1-(1-p)^{\binom{r}{2}}} &\le \frac{\left(\frac{ne}{r}\right)^r}{\left(\frac{1}{1-p}\right)^{\binom{r}{2}}}(1+o(1)) = \left(\frac{ne}{r\left(\frac{1}{1-p}\right)^{\frac{r-1}{2}}}\right)^r(1+o(1)) \\
&= \left(\frac{ne}{r\left(1+\frac{p}{1-p}\right)^{\frac{r-1}{2}}}\right)^r(1+o(1)) \\
&\le \left(\frac{ne}{r\exp\left(\frac{\frac{p}{1-p}}{1+\frac{p}{1-p}}\cdot\frac{r-1}{2}\right)}\right)^r(1+o(1)) \\
&= \left(\frac{ne}{r\exp\left(p\cdot\frac{r-1}{2}\right)}\right)^r(1+o(1)) \\
&= \left(\frac{ne^{1+\frac{p}{2}}}{re^{\frac{rp}{2}}}\right)^r(1+o(1)) \qquad (6.9) \\
&= \left(\frac{e^{1+\frac{p}{2}}}{r}\right)^r(1+o(1)) \\
&\le \left(\frac{e^{\frac{5}{4}}}{r}\right)^r(1+o(1)). \qquad (6.10)
\end{aligned}$$

The quantity $\left(\frac{e^{\frac{5}{4}}}{r}\right)^r$ in Eq. (6.10) is less than 0.5^r when n is sufficiently large, the latter approaches 0 when $n \to +\infty$. This proves that when $p \le \frac{1}{2}$, $\Pr[\alpha(\mathfrak{g}(G,p)) > r] \to 0$ as $n \to +\infty$. That is, when $p \le \frac{1}{2}$, a.a.s. $\alpha(\mathfrak{g}(G,p)) \le 2\ln n/p$.

When $p \ge \frac{1}{2}$, that is $a = \frac{p}{1-p} \ge 1$, $q = 1-p \le \frac{1}{2}$ and exactly the same argument as when $p \le \frac{1}{2}$ applies by replacing p with $1-q$, which shows that a.a.s. $\alpha(\mathfrak{g}(G,p)) \le 2\ln n/(1-p)$. This proves the theorem. □

Since $\alpha(\mathfrak{g}(G,p)) \geq i(\mathfrak{g}(G,p))$, $\Pr[i(\mathfrak{g}(G,p)) > r] \leq \Pr[\alpha(\mathfrak{g}(G,p)) > r]$ and thus we have the following corollary:

Corollary 6.7. *For any graph $G = (V,E)$, a.a.s.*

$$i(\mathfrak{g}(G,p)) \leq \begin{cases} \frac{2\ln n}{p}, & \text{if } p \in (\frac{2\ln n}{n}, \frac{1}{2}], \\ \frac{2\ln n}{1-p}, & \text{if } p \in [\frac{1}{2}, 1 - \frac{2\ln n}{n}). \end{cases}$$

6.3 A tail bound on the independent domination number

Theorem 6.8. *For any graph $G = (V,E)$ and $p \in (\frac{4\ln^2 n}{n}, \frac{1}{2}]$,*

$$\Pr[i(\mathfrak{g}(G,p)) \geq \sqrt{\frac{4n}{p}}] \leq \Pr[\alpha(\mathfrak{g}(G,p)) \geq \sqrt{\frac{4n}{p}}] \leq 2^{-n}.$$

Proof. The proof of this theorem flows exactly the same as the proof of Theorem 6.6. In fact, with $p \leq \frac{1}{2}$, we have both Eq. (6.6) and Eq. (6.7) hold. Different from the proof of Theorem 6.6 where $r = 2\ln n/p$, we have now $r = \sqrt{\frac{4n}{p}} \geq 2\ln n/p$ and therefore Eq. (6.8) holds as well. Again, using Eq. (6.8) and Fact 6.1, when n is sufficiently large, Eq. (6.9) still holds. It then follows from Fact 6.1 that Eq. (6.6) becomes

$$\begin{aligned} \Pr[i(\mathfrak{g}(G,p)) \geq r] &\leq \Pr[\alpha(\mathfrak{g}(G,p)) \geq r] \\ &\leq 1 - \exp\left(-\left(\frac{ne^{1+\frac{p}{2}}}{re^{\frac{rp}{2}}}\right)^r (1+o(1))\right). \end{aligned} \tag{6.11}$$

Using $r = \sqrt{\frac{4n}{p}}$, we prove in the following that $\left(\frac{ne^{1+\frac{p}{2}}}{re^{\frac{rp}{2}}}\right)^r (1+o(1)) = o(1)$. And consequently by Fact 6.1 again and $r = \sqrt{\frac{4n}{p}} \geq \sqrt{8n}$, Eq. (6.11) becomes

$$\begin{aligned} \Pr[i(\mathfrak{g}(G,p)) \geq r] &\leq \left(\frac{ne^{1+\frac{p}{2}}}{re^{\frac{rp}{2}}}\right)^r (1+o(1)) \\ &\leq \frac{e}{2}\left(\frac{ne^{1+\frac{p}{2}}}{re^{\frac{rp}{2}}}\right)^r \\ &= \frac{e}{2}\exp\left(-r\left(\ln r + \frac{1}{2}rp - \ln n - 1 - \frac{p}{2}\right)\right) \\ &= \frac{e}{2}\exp\left(-r\left(\ln r + \frac{1}{4}rp - \ln n - 1 - \frac{p}{2}\right) - \frac{1}{4}r^2p\right) \\ &= \frac{e}{2}\exp\left(-r\left(\ln r + \frac{1}{4}rp - \ln n - 1 - \frac{p}{2}\right) - n\right). \end{aligned} \tag{6.12}$$

The quantity $\left(\ln r + \frac{1}{4}rp - \ln n - 1 - \frac{p}{2}\right)$ in Eq. (6.12) is non-negative when $n \geq 2$, since

$$\begin{aligned}
\ln r + \frac{1}{4}rp - \ln n - 1 - \frac{p}{2} &\geq \frac{1}{2}\ln(8n) + \frac{1}{4}\sqrt{4np} - \ln n - 1 - \frac{1}{4} \\
&\geq \frac{1}{2}\ln(8n) + \frac{1}{4}\sqrt{4n \cdot \frac{4\ln^2 n}{n}} - \ln n - 1 - \frac{1}{4} \\
&= \frac{1}{2}\left(\ln(8n) - \frac{5}{2}\right) \\
&\geq 0.
\end{aligned}$$

It follows that Eq. (6.12) becomes

$$\begin{aligned}
\Pr[i(\mathfrak{g}(G,p)) \geq r] &\leq \frac{e}{2}\exp\left(-r\left(\ln r + \frac{1}{4}rp - \ln n - 1 - \frac{p}{2}\right) - n\right) \\
&\leq \frac{e}{2}\, e^{-n} \\
&< 2^{-n}.
\end{aligned}$$

This proves the theorem. □

6.4 Approximating the independent domination number

We present next a simple algorithm, denoted as *Approx-IDS*, for computing an independent dominating set in $\mathfrak{g}(G,p)$. In the first phase, algorithm Approx-IDS repeatedly picks a maximum degree vertex and updates the graph by deleting the picked vertex and all its neighbors; it terminates when there is no more vertex and returns a subset I of V. If $|I| \leq \sqrt{\frac{4n}{p}}$, algorithm Approx-IDS terminates and outputs I; otherwise it moves into the second phase. In the second phase, algorithm Approx-IDS performs an exhaustive search over all subsets of V, and returns the minimum independent dominating set I^*.

Theorem 6.9. *For any graph $G = (V,E)$ and $p \in (\frac{4\ln^2 n}{n}, \frac{1}{2}]$, algorithm* Approx-IDS *is a $\sqrt{\frac{4n}{p}}$-approximation to the* MIDS *problem on the perturbed graph $\mathfrak{g}(G,p)$, and it has polynomial expected running time.*

Proof. Note that $i(\mathfrak{g}(G,p)) \geq 1$. The subset I of V computed by algorithm Approx-IDS is a dominating set, since every vertex of V is either in I, or is a neighbor of some vertex in I. Also, no two vertices of I can be adjacent, since otherwise one would be removed in the iteration its neighbor was picked by the algorithm. Therefore, I is an independent dominating set of $\mathfrak{g}(G,p)$. It follows that if algorithm Approx-IDS terminates after the first phase, $|I| \leq \sqrt{\frac{4n}{p}} \cdot i(\mathfrak{g}(G,p))$. Also clearly the first phase takes $O(n^3)$ time.

In the second phase, a maximum of 2^n subsets of V are examined by the algorithm. Since checking each of them to be an independent dominating set or not takes no more than $O(n^2)$ time, the overall running time is $O(2^n n^2)$. Note that this phase returns I^* with $|I^*| = i(\mathfrak{g}(G,p))$. As $\alpha(\mathfrak{g}(G,p)) \geq |I| > \sqrt{\frac{4n}{p}}$, Theorem 6.8 tells that the probability of executing this second phase is no more than 2^{-n}. Therefore, the expected running time of the second phase is $O(n^2)$. This proves the theorem. □

6.5 Conclusions and future work

We performed a probabilistic analysis of the approximability of the minimum independent dominating set problem and proposed an $O(\sqrt{\frac{4n}{p}})$-approximation algorithm with polynomial expected running time. Our version of the minimum independent dominating set problem is a classic version, where each vertex has an unit weight. It would be interesting to conduct a smoothed analysis for the weighted minimum independent dominating set problem under the perturbation model which perturbs the weight of each vertex.

Chapter 7

Trie and Patricia Index Trees[1]

7.1 Introduction

A *Trie*, also known as a *digital tree*, is an ordered tree data structure for storing strings over an alphabet Σ. It was initially developed and analyzed by Fredkin [42] in 1960 and Knuth [61] in 1973. Such a data structure is used for storing a dynamic set to be exploited as an associative array, where keys are strings. There has been much recent exploitation of such index trees for processing genomic data.

In the simplest form, let the alphabet be $\Sigma = \{0, 1\}$ and consider a set $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ of n binary strings over Σ, where each s_i can be infinitely long. The Trie for storing these n binary strings is an ordered binary tree $T_{\mathcal{S}}$: first, each s_i defines a path (infinite if its length $|s_i|$ is infinite) in the tree, starting from the root, such that a 0 forces a move to the left and a 1 indicates a move to the right; if one node is the highest in the tree that is passed through by only one string $s_i \in \mathcal{S}$, then the path defined by s_i is truncated at this node, which becomes a leaf in the tree and is associated (*i.e.*, labelled) with s_i. The *height* of the Trie $T_{\mathcal{S}}$ built over $\mathcal{S}$ is defined as the number of edges on the longest root-to-leaf path. FIGURE 7.1 shows the Trie constructed for a set of six strings. (These strings can be long or even infinite, but only the first 5 bits are shown, which are those used in the example construction.)

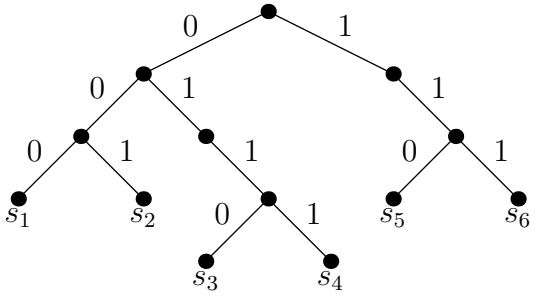

FIGURE 7.1: The Trie constructed for $\{s_1 = 00001\ldots,\ s_2 = 00111\ldots,\ s_3 = 01100\ldots,\ s_4 = 01111\ldots,\ s_5 = 11010\ldots,\ s_6 = 11111\ldots\}$.

[1]This chapter is based on [102].

Let H_n denote the height of the Trie on a set of n binary strings. It is not hard to see that in the worst case H_n is unbounded, due to the existence of two of the strings sharing an arbitrary long common prefix. In the uniform distribution model, bits of s_i are *independent and identically distributed* (*i.i.d.*) Bernoulli random variables each of which takes 1 with probability $p = 0.5$. The asymptotic behavior of Trie height H_n under the uniform distribution model had been well studied in the 1980s [36, 40, 41, 69, 80, 81, 83, 95, 96], and it is known that *asymptotically almost surely* (*a.a.s.*)

$$H_n / \log_2 n \to 2, \text{ when } n \to \infty.$$

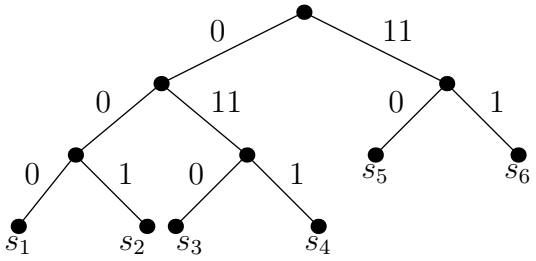

FIGURE 7.2: The Patricia constructed for $\{s_1 = 00001\ldots, s_2 = 00111\ldots, s_3 = 01100\ldots, s_4 = 01111\ldots, s_5 = 11010\ldots, s_6 = 11111\ldots\}$.

A Patricia index tree is a space-optimized variant of the Trie data structure, in which every node with only one child is merged with its child. Such a data structure was firstly discovered by Morrison [72] in 1968, and then well analyzed in "The art of computer programming" by Knuth [61] in 1973. FIGURE 7.2 shows the Patricia tree constructed for the same set of six strings used in FIGURE 7.1. Again let H_n denote the height of the Patricia tree on a set of n binary strings. In the worst case, $H_n = n - 1$, where s_i is in the form $1 \ldots 100 \ldots$ with a prefix consisting of $i - 1$ consecutive 1's. Under the same uniform distribution model assumed for an average case analysis on Trie height, Pittel showed that a.a.s. the height of Patricia is only 50% of the height of Trie [80], that is,

$$H_n / \log_2 n \to 1, \text{ when } n \to \infty.$$

The average case analysis is intended to provide insight on the practical performance as a string indexing structure. In 2002, Nilsson and Tikkanen [77] experimentally investigated the height of Patricia trees and other search structures. In particular, they showed that the heights of the Patricia trees on sets of $50,000$ random uniformly distributed strings are 15.9 on average and 20 at most. For real datasets consisting of $19,461$ strings from geometric data on drill holes, $16,542$ ASCII character strings from a book, and $38,367$ strings from Internet routing tables, the heights of the Patricia trees are on average 20.8, 20.2, 18.6, respectively, and at most 30, 41, 24, respectively.

Theoretically speaking, these experimental results suggest that worst-case instances are perhaps only isolated peaks in the instance space. This hypothesis is partially supported by the average case analysis on the heights of Trie and Patricia structures, under the uniform distribution model, that suggests the heights are *a.a.s.* logarithmic. Nevertheless, these average case analysis results on the specific random instances generated under the uniform distribution model could be inconclusive, because the specific random instances have very special properties inherited from the model, and thus would distinguish themselves from real-world instances.

We conduct the smoothed analysis on the heights of Trie and Patricia index trees, to reveal certain essential properties of these two data structures. We first introduce the string perturbation model, and show an *a.a.s.* upper bound $O(\log n)$ and an *a.a.s.* lower bound $\Omega(\log n)$ on the Trie height H_n. The consequence is that the smoothed height of the Trie on n strings is in $\Theta(\log n)$. Then, we achieve similar results for the smoothed height of the Patricia tree on n strings, that is, $H_n = \Theta(\log n)$, which explains the practical performance of Patricia in the experiments conducted by Nilsson and Tikkanen [77].

7.2 The smoothed height of Trie

We consider an arbitrary set $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ of n strings over alphabet $\{0, 1\}$, where each string may be infinitely long. Let $s_i(\ell)$ denote the ℓ-th bit in string s_i, for $i = 1, 2, \ldots, n$ and $\ell = 1, 2, 3, \ldots$. Every string s_i is perturbed by adding a noise string ν_i, giving rise to the perturbed string $\tilde{s}_i = s_i + \nu_i$, where $\tilde{s}_i(\ell) = s_i(\ell)$ if and only if $\nu_i(\ell) = 0$. The noise string ν_i is independently generated by a memoryless source, which assigns 1 to every bit of string ν_i independently and with a small probability $\epsilon \in [0, 0.5]$. More formally,

$$Pr\{\nu_i(\ell) = 1\} = \epsilon \text{ for each } \ell = 1, 2, 3, \ldots.$$

Essentially the perturbation flips each bit of every string independently and with a probability ϵ. Let $\tilde{\mathcal{S}} = \{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n\}$ denote the set of perturbed strings.

Let p_{ij}^{ℓ} be the probability of the event $\{\tilde{s}_i(\ell) = \tilde{s}_j(\ell)\}$. We have

$$p_{ij}^{\ell} = \begin{cases} 2\epsilon(1-\epsilon) \stackrel{\triangle}{=} p, & \text{if } s_i(\ell) \neq s_j(\ell), \\ \epsilon^2 + (1-\epsilon)^2 = 1 - p \stackrel{\triangle}{=} q, & \text{if } s_i(\ell) = s_j(\ell). \end{cases} \tag{7.1}$$

We can clearly note that $q \geq p$, since $\epsilon \leq 0.5$. Let C_{ij} denote the length of the longest common prefix between $\tilde{s}_i$ and $\tilde{s}_j$. Since $C_{ij} = k$ if and only if $\tilde{s}_i(\ell) = \tilde{s}_j(\ell)$

for $\ell = 1, 2, \ldots, k$ but not for $\ell = k+1$, the probability of $\{C_{ij} = k\}$ for any $k \geq 0$ is

$$Pr\{C_{ij} = k\} = \left(\prod_{\ell=1}^{k} p_{ij}^{\ell}\right)(1 - p_{ij}^{k+1}).$$

From the fact that $\{C_{ij} = k\}$ and $\{C_{ij} = m\}$ are disjoint events when $k \neq m$, we have for any $k \geq 1$

$$Pr\{C_{ij} < k\} = \sum_{m=0}^{k-1} \left(\prod_{\ell=1}^{m} p_{ij}^{\ell} - \prod_{\ell=1}^{m+1} p_{ij}^{\ell}\right) = 1 - \prod_{\ell=1}^{k} p_{ij}^{\ell}.$$

Consequently, the probability that the longest common prefix between $\tilde{s}_i$ and $\tilde{s}_j$ is at least k long is

$$Pr\{C_{ij} \geq k\} = 1 - Pr\{C_{ij} < k\} = \prod_{\ell=1}^{k} p_{ij}^{\ell}. \tag{7.2}$$

7.2.1 An *a.a.s.* upper bound

We use a slight abuse of notation H_n to also denote the height of the Trie constructed for $\tilde{\mathcal{S}}$. We can express H_n in terms of C_{ij} as

$$H_n = \max_{1 \leq i < j \leq n} C_{ij} + 1.$$

By Boole inequality [32], we have

$$\begin{aligned} Pr\{H_n > k\} &= Pr\{\max_{1 \leq i < j \leq n} C_{ij} \geq k\} \\ &\leq \binom{n}{2} \prod_{\ell=1}^{k} p_{ij}^{\ell} \leq \binom{n}{2} q^k, \end{aligned}$$

where the last equality holds when all the n strings $\{s_1, s_2, \ldots, s_n\}$ have the same prefix of length k. By setting $k = 2(1+\delta)\log_{1/q} n$ for a constant $\delta > 0$, we have

$$Pr\{H_n > k\} \leq \binom{n}{2} q^{2(1+\delta)\log_{1/q} n} \leq n^{-2\delta} \to 0,$$

as $n \to \infty$. Therefore, $H_n \leq 2\log_{1/q} n$ with high probability, when n approaches infinity.

7.2.2 An *a.a.s.* lower bound

To estimate a lower bound, we will use the following Chunge-Erdös formulation of the second moment method on a set of events:

Lemma 7.1. (Chunge-Erdös) [31] *For any set of events* $E_1, E_2, \ldots, E_n$,

$$Pr\{\cup_{i=1}^{n} E_i\} \geq \frac{(\sum_{i=1}^{n} Pr\{E_i\})^2}{\sum_{i=1}^{n} Pr\{E_i\} + \sum_{i \neq j} Pr\{E_i \cap E_j\}}.$$

Let A_{ij} denote the event $\{C_{ij} \geq k\}$, for every pair $\{i, j\}$ such that $1 \leq i < j \leq n$; also define the following two sums:

$$S_1 \stackrel{\triangle}{=} \sum_{1 \leq i < j \leq n} Pr\{A_{ij}\}, \text{ and}$$
$$S_2 \stackrel{\triangle}{=} \sum_{\{i,j\} \neq \{s,t\}} Pr\{A_{ij} \cap A_{st}\}.$$

Then by Chunge-Erdös formulation (Lemma 7.1), we have

$$Pr\{H_n > k\} = Pr\{\cup_{1 \leq i < j \leq n} A_{ij}\} \geq \frac{S_1^2}{S_1 + S_2}. \tag{7.3}$$

Let's first estimate S_1. From Eq. (7.2), one clearly sees that

$$S_1 = \sum_{1 \leq i < j \leq n} Pr\{A_{ij}\} = \sum_{1 \leq i < j \leq n} \prod_{\ell=1}^{k} p_{ij}^{\ell}. \tag{7.4}$$

Recall the definition of p_{ij}^{ℓ} and its value in Eq. (7.1). The following Lemma 7.2 is then straight-forward:

Lemma 7.2. *For any* $\ell \geq 1$ *and any three perturbed strings* $\tilde{s}_i, \tilde{s}_j, \tilde{s}_t$, *if* $p_{ij}^{\ell} = p_{it}^{\ell}$, *then* $p_{jt}^{\ell} = q$.

Lemma 7.3. *For any three perturbed strings* $\tilde{s}_i, \tilde{s}_j, \tilde{s}_t$,

$$S_0 \stackrel{\triangle}{=} \prod_{\ell=1}^{k} p_{ij}^{\ell} + \prod_{\ell=1}^{k} p_{it}^{\ell} + \prod_{\ell=1}^{k} p_{jt}^{\ell} \geq 3p^{\frac{2}{3}k} q^{\frac{1}{3}k}.$$

Proof. For the string pair (s_i, s_j), let Z_{ij} denote the number of $(0,1)$-pairs and $(1,0)$-pairs in $\{(s_i(\ell), s_j(\ell)), 1 \leq \ell \leq k\}$, that is, the number of bits where s_i and s_j have different values among the first k bits. Clearly from Eq. (7.1),

$$\prod_{\ell=1}^{k} p_{ij}^{\ell} = p^{Z_{ij}} q^{k - Z_{ij}}.$$

For the string triple (s_i, s_j, s_t), let x_{ij} denote the number of $(0,0,1)$-triples and $(1,1,0)$-triples in $\{(s_i(\ell), s_j(\ell), s_t(\ell)), 1 \leq \ell \leq k\}$; likewise, x_{it} and x_{jt} are similarly defined. Also

let y denote the number of $(0,0,0)$-triples and $(1,1,1)$-triples in $\{(s_i(\ell), s_j(\ell), s_t(\ell)), 1 \leq \ell \leq k\}$. The following relationships are direct consequences of the definitions:

$$\begin{aligned} Z_{ij} &= x_{it} + x_{jt}, \\ Z_{it} &= x_{ij} + x_{jt}, \\ Z_{jt} &= x_{ij} + x_{it}, \\ k &= x_{ij} + x_{it} + x_{jt} + y. \end{aligned}$$

It follows that

$$\begin{aligned} S_0 &\stackrel{\triangle}{=} \prod_{\ell=1}^{k} p_{ij}^{\ell} + \prod_{\ell=1}^{k} p_{it}^{\ell} + \prod_{\ell=1}^{k} p_{jt}^{\ell} \\ &= p^{x_{it}+x_{jt}} q^{x_{ij}+y} + p^{x_{ij}+x_{jt}} q^{x_{it}+y} + p^{x_{ij}+x_{it}} q^{x_{jt}+y} \\ &= p^k \left[\left(\frac{q}{p}\right)^{x_{ij}+y} + \left(\frac{q}{p}\right)^{x_{it}+y} + \left(\frac{q}{p}\right)^{x_{jt}+y} \right]. \end{aligned}$$

One can check that, since $q \geq p$, the quantity in the last line reaches the minimum when $x_{ij} = x_{it} = x_{jt} = k/3$ and $y = 0$. That is,

$$S_0 \stackrel{\triangle}{=} \prod_{\ell=1}^{k} p_{ij}^{\ell} + \prod_{\ell=1}^{k} p_{it}^{\ell} + \prod_{\ell=1}^{k} p_{jt}^{\ell} \geq 3p^{\frac{2}{3}k} q^{\frac{1}{3}k}.$$

This proves the lemma. □

Note that each string pair (s_i, s_j) is involved in exactly $n-2$ string triples (s_i, s_j, s_t), for $t \neq i, j$. By Lemma 7.3, Eq. (7.4) becomes

$$\begin{aligned} S_1 &= \sum_{1 \leq i < j \leq n} \prod_{\ell=1}^{k} p_{ij}^{\ell} \\ &\geq \frac{1}{n-2} \binom{n}{3} 3p^{\frac{2}{3}k} q^{\frac{1}{3}k} \\ &= \binom{n}{2} p^{\frac{2}{3}k} q^{\frac{1}{3}k}. \end{aligned} \tag{7.5}$$

We next estimate S_2, which is a bit harder because two events A_{ij} and A_{st} may not be independent. We split S_2 into two parts: $S_2 = S_2' + S_2''$, where

$$S_2' \stackrel{\triangle}{=} \sum_{\{i,j\} \cap \{s,t\} = \emptyset} Pr\{A_{ij} \cap A_{st}\}, \text{ and}$$

$$S_2'' \stackrel{\triangle}{=} \sum_{\{i,j\} \cap \{s,t\} \neq \emptyset} Pr\{A_{ij} \cap A_{st}\}.$$

Since two events C_{ij} and C_{st} are independent when $\{i,j\} \cap \{s,t\} = \emptyset$, we can estimate S_2' as follows:

$$\begin{aligned} S_2' &= \sum_{\{i,j\}\cap\{s,t\}=\emptyset} \Big(Pr\{A_{ij}\} Pr\{A_{st}\} \Big) \\ &\leq \left(\sum_{\{i,j\}} Pr\{A_{ij}\} \right)^2 = S_1^2. \end{aligned}$$

Event $\{A_{ij} \cap A_{it}\}$ is equivalent to the event in which the first k bits of all three perturbed strings $\tilde{s}_i, \tilde{s}_j$, and $\tilde{s}_t$ are identical. Using $\epsilon \leq 0.5$, we have

$$\begin{aligned} Pr\{A_{ij} \cap A_{it}\} &= Pr\{\tilde{s}_i(\ell) = \tilde{s}_j(\ell) = \tilde{s}_t(\ell), 1 \leq \ell \leq k\} \\ &\leq \Big(\epsilon^3 + (1-\epsilon)^3 \Big)^k. \end{aligned}$$

It follows that

$$\begin{aligned} S_2'' &= \sum_{\{i,j\}\cap\{s,t\}\neq\emptyset} Pr\{A_{ij} \cap A_{st}\} \\ &\leq 3\binom{n}{3} \Big(\epsilon^3 + (1-\epsilon)^3 \Big)^k \leq 3\binom{n}{3}, \end{aligned}$$

where the factor 3 arises because a string triple $\{\tilde{s}_i, \tilde{s}_j, \tilde{s}_t\}$ gives rise to three events $\{A_{ij} \cap A_{it}\}$, $\{A_{ij} \cap A_{jt}\}$, and $\{A_{it} \cap A_{jt}\}$.

Putting S_2' and S_2'' together, we can upper bound S_2 by

$$S_2 = S_2' + S_2'' \leq S_1^2 + 3\binom{n}{3}. \tag{7.6}$$

Using the estimates of S_1 and S_2 in Eqs. (7.5) and (7.6) respectively, Eq. (7.3) becomes

$$\begin{aligned} Pr\{H_n > k\} &\geq \frac{S_1^2}{S_1 + S_2} \\ &= \frac{1}{1/S_1 + (S_2' + S_2'')/S_1^2} \\ &\geq \frac{1}{1/S_1 + 1 + S_2''/S_1^2} \\ &\geq \frac{1}{1 + \frac{1}{\binom{n}{2} p^{\frac{2}{3}k} q^{\frac{1}{3}k}} + \frac{3\binom{n}{3}}{\left(\binom{n}{2} p^{\frac{2}{3}k} q^{\frac{1}{3}k}\right)^2}} \\ &\geq \frac{1}{1 + 4n^{-2} p^{-\frac{2}{3}k} q^{-\frac{1}{3}k} + 2n^{-1} p^{-\frac{4}{3}k} q^{-\frac{2}{3}k}} \end{aligned}$$

$$\begin{aligned} &\geq \frac{1}{1+4n^{-2}n^{2(1-\delta)}+2n^{-1}n^{1-\delta}} \qquad (7.7)\\ &= \frac{1}{1+4n^{-2\delta}+2n^{-\delta}}\\ &\geq 1-O(n^{-\delta}) \to 1, \end{aligned}$$

where the inequality Eq. (7.7) is achieved by setting

$$k = 2(1-\delta)\log_{p^{-2/3}q^{-1/3}} n, \text{ that is, } p^{-\frac{2}{3}k}q^{-\frac{1}{3}k} = n^{2(1-\delta)},$$

for a constant $\delta > 0$. Therefore, H_n is larger than $2\log_{p^{-2/3}q^{-1/3}} n$ with a high probability when n approaches infinity.

Theorem 7.4. *The smoothed height of the Trie on n strings is in $\Theta(\log n)$, where the bit perturbation model is* i.i.d. *Bernoulli distribution.*

7.3 The smoothed height of Patricia

Here we briefly do the smoothed analysis on the height of the Patricia tree on a set of n binary strings. We adopt the same *i.i.d.* Bernoulli bit perturbation model as in the last section. Again, we present an *a.a.s.* upper bound and an *a.a.s.* lower bound for the smoothed height.

7.3.1 An *a.a.s.* upper bound

Following Pittel [80], on the set of n perturbed strings $\tilde{\mathcal{S}} = \{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n\}$, we claim that for any fixed integers $k \geq 0$ and $b \geq 2$, the event $\{H_n \geq k+b-1\}$ implies the event that there exist b strings $\tilde{s}_{i_1}, \tilde{s}_{i_2}, \ldots, \tilde{s}_{i_b}$ such that their common prefix is of length at least k (denoted as $C_{i_1 i_2 \ldots i_b} \geq k$). The correctness of the above claim follows from because, in Patricia trees, there are no degree-2 nodes (except for the root), and thus a path of length $k+b-1$ hints at least b leaves in the subtree rooted at the node at distance k from the Patricia root.

Similar to the definition of p^{ℓ}_{ij} in Eq. (7.1), $p^{\ell}_{i_1 i_2 \ldots i_b}$ denotes the probability of the event $\{\tilde{s}^{\ell}_{i_1} = \tilde{s}^{\ell}_{i_2} = \ldots = \tilde{s}^{\ell}_{i_b}\}$, for any $b \geq 2$, which is calculated as follows:

$$p^{\ell}_{i_1 i_2 \ldots i_b} = (1-\epsilon)^{k_0}\epsilon^{k_1} + (1-\epsilon)^{k_1}\epsilon^{k_0},$$

where k_0 and k_1 are the number of 0's and 1's among the b bit values $\tilde{s}_{i_1}(\ell), \tilde{s}_{i_2}(\ell), \ldots, \tilde{s}_{i_b}(\ell)$, respectively. By a similar argument as presented for $Pr\{A_{ij}\}$ in Section 2, we have

$$Pr\{C_{i_1 i_2 \ldots i_b} \geq k\} = \prod_{\ell=1}^{k} p_{i_1 i_2 \ldots i_b}^{\ell}.$$

For a fixed $b \geq 2$, let $q_b = \epsilon^b + (1-\epsilon)^b$ and $k = k_b = b(1+\delta/2)\log_{1/q_b} n$. We have

$$\begin{aligned} k &= b(1+\delta/2)\log_{1/q_b} n \\ &= (1+\delta/2)\frac{\ln n}{\ln q_b^{-1/b}} \\ &= (1+\delta/2)\frac{\ln n}{\ln \left(\epsilon^b + (1-\epsilon)^b\right)^{-1/b}} \\ &\leq (1+\delta/2)\frac{\ln n}{\ln \left(\epsilon^2 + (1-\epsilon)^2\right)^{-1/2}} \qquad (7.8) \\ &= 2(1+\delta/2)\log_{1/q} n, \end{aligned}$$

where the inequality in Eq. (7.8) holds for any $b \geq 2$. Setting $b = \delta \log_{1/q} n$, it follows that

$$\begin{aligned} Pr\{H_n \geq 2(1+\delta)\log_{1/q} n\} &\leq Pr\{H_n \geq k+b-1\} \\ &\leq Pr\{\max_{i_1,i_2,\ldots,i_b} C_{i_1 i_2 \ldots i_b} \geq k\} \\ &\leq n^b \prod_{\ell=1}^{k} p_{i_1 i_2 \ldots i_b}^{\ell} \\ &\leq n^b q_b^k \\ &\in O(n^{-b\delta}) \to 0, \end{aligned}$$

when $n \to \infty$.

In summary, for any $\delta > 0$, we have

$$Pr\{H_n \geq 2(1+\delta)\log_{1/q} n\} \in O(n^{-b\delta}) \to 0,$$

when n approaches infinity, and thus *a.a.s.* $H_n \leq 2(1+\delta)\log_{1/q} n$.

7.3.2 An *a.a.s.* lower bound

Let D_i be the depth of node labelled $\tilde{s}_i$ in the Patricia tree.

Clearly, $H_n = \max_{i=1}^{n} D_i$ and the $\tilde{s}_{i^*}$ reaching the maximum depth must be a leaf node. It follows that if $H_n < k$, then at least one of the 2^k possible length-k strings does not appear as a prefix of any perturbed strings $\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n$.

Let $\mathbb{L}n = \log_{1/\epsilon} n$ and $k = \mathbb{L}\frac{n}{\mathbb{L}\ln n}$. We have

$$\begin{aligned} Pr\{H_n < k\} &\leq 2^k Pr\{\text{no } \tilde{s}_i \text{ starts with } k \text{ 0's}\} \\ &\leq 2^k (1-\epsilon^k)^n \\ &\leq 2^k e^{-\epsilon^k n} \\ &= \exp\{k \ln 2 - \epsilon^k n\} \\ &= \exp\{\ln 2 \cdot \mathbb{L}\frac{n}{\mathbb{L}\ln n} - \mathbb{L}\ln n\} \to 0, \end{aligned}$$

when n approaches infinity, and thus *a.a.s.* $H_n \geq \mathbb{L}\frac{n}{\mathbb{L}\ln n}$.

In summary, we have the following theorem.

Theorem 7.5. *The smoothed height of the Patricia on n strings is in $\Theta(\log n)$, where the bit perturbation model is* i.i.d. *Bernoulli distribution.*

7.4 Conclusions and future work

We conduct the smoothed analysis on the heights of Trie and Patricia index trees, to reveal certain essential properties of these two data structures. We showed that the smoothed height of the Trie on n strings is in $\Theta(\log n)$. And we achieved a similar result for the smoothed height of the Patricia tree on n strings, that is, $H_n = \Theta(\log n)$, which explains the practical performance of Patricia in the experiments conducted by Nilsson and Tikkanen [77]. In our string perturbation model, we assume the perturbation for each position is independent. It would be interesting to investigate these two data structures under a new string perturbation model, which considers the dependence of the noise added to every pair of positions.

Chapter 8

Conclusions and Future Work

8.1 Summary

How to evaluate the performance of an algorithm is a very important subject in computer science, for understanding its applicability, for understanding the problem to which it is applied, and for the development of new ideas that help to improve the existing algorithms. There are two main factors, *i.e.* the performance measure and the analysis model, that affect the evaluation of an algorithm.

The performance of an algorithm can be evaluated by many performance measures. Usually, these measures are time-related or space-related. The ideal case is that an algorithm can always return an optimal solution to a problem while the time and space consumed are within an appropriate tolerance. Unfortunately, most interesting optimization problems raising from the real world applications are NP-hard. For these problems, we need to look for trade-offs between the qualities of the solution and the running time (or space consumption) of the algorithm, and thus approximation algorithms would attract more interests.

For the analysis models, there are two classic analysis approaches, *i.e.* the worst-case and the average-case analyses. An algorithm with good worst-case performance is very desirable because it performs well on all possible inputs. However, a bad worst-case performance does not necessarily imply that the algorithm performs also badly in practice. It might be the case that the "hard" instances are relatively "isolated" in the instance space. This motivates to study the average-case performance rather than the worst-case performance. But the average-case analysis is often problematic because it is not clear how to choose a "reasonable" probability distribution on the set of inputs and thus most average-case analyses assume simple distributions instead, which make the analysed instances do not reflect typical instances. The smoothed analysis circumvents the drawbacks of worst-case and average-case analyses. It can not only rule out artificial worst-case instances by the random perturbation, but also prevent the analysed instances dominated by completely random instances since the adversary can approximately determine the structure of the instance. By defining the perturbation appropriately, the

smoothed performance would be more "realistic". Since the smoothed analysis was introduced by Spielman and Teng in 2001, it has achieved a series of successes on running time analysis for many the most interested algorithms.

In this thesis, we concentrated on the analysis of approximation ratios for the approximation algorithms under the worst-case analysis and the smoothed analysis. In particular, we designed currently the best approximation algorithms for several interesting NP-hard problems from bioinformatics, networking and graph theory.

8.2 Future work

Though there are several commonly used perturbation models as we introduced in Chapter 1, it is hard to define a reasonable one for most discrete combinatorial optimization problems. For example, for the minimum independent dominating set problem in Chapter 6, the perturbed graph is defined based on a simple extension of the Erdös–Rényi model. Such a perturbation model indeed introduces a small amount of randomness (or noise) by the intuition that an edge will be presented in the perturbed graph with high (low, respectively) probability if it is (isn't, respectively) originally in the given graph. Nevertheless, in real application this might be unreasonable as the perturbation for each edge might not be independent. Therefore to define more reasonable perturbation models are quite challenging for the future work.

For an approximation algorithm, we usually concentrate on the quality of the solution it returns. As far as we know, there are very few papers studying the smoothed analysis on approximation ratio. We guess it is mainly due to the difficulty to estimate the optimal solution, given that it is already very challenging to estimate an optimal solution under the average-case analysis model, which is much "weaker" than the smoothed analysis. However, in real world applications, lots of approximation algorithms perform very well in practice but have poor approximation ratios under the worst-case analysis. Therefore, smoothed analysis on the performance ratio of approximation algorithms is of great significance and would help us to understand these algorithms better. Besides, for a certain problem, there may exist some quantity that reveals some essential properties of the problem itself, which in turn may help us to better understand the problem or to design new more efficient and more effective algorithms. Thus measuring such quantities under the smoothed analysis would also be of great significance. What's more, it would be interesting to use these smoothed measures to depict the instance space in the future research work.

Bibliography

[1] S. Ahmed, S. Mneimneh, and N. Greenbaum. A combinatorial approach for multiple RNA interaction: formulations, approximations, and heuristics. In *COCOON*, LNCS 7936, pages 421–433, 2013.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithm, and Applications*. China Machine Press, 2005.

[3] C. Alkan, E. Karakoç, J. H. Nadeau, S. C. Sahinalp, and K. Zhang. RNA-RNA interaction prediction and antisense RNA target search. *Journal of Computational Biology*, 13:267–282, 2006.

[4] G. E. Andrews. *The Theory of Partitions*. Addison-Wesley, 1976.

[5] G. E Andrews and K. Eriksson. *Integer Partitions*. Cambridge University Press, 2004.

[6] S. Angibaud, G. Fertin, I. Rusu, A. Thévenin, and S. Vialette. On the approximability of comparing genomes with duplicates. *Journal of Graph Algorithms and Applications*, 13:19–53, 2009.

[7] Richard P. Anstee. A polynomial algorithm for b-matchings: an alternative approach. *Information Processing Letters*, 24:153–157, 1987.

[8] E. M. Arkin and R. Hassin. On local search for weighted k-set packing. *Mathematics of Operations Research*, 23:640–648, 1998.

[9] D. A. Babayev, G. I. Bell, and U. G. Nuriyev. The Bandpass problem: combinatorial optimization and library of problems. *Journal of Combinatorial Optimization*, 18:151–172, 2009.

[10] R. Beier and B. Vöcking. Random knapsack in expected polynomial time. In *STOC*, pages 232–241, 2003.

[11] G. I. Bell and D. A. Babayev. Bandpass problem. In *Annual INFORMS meeting*, Denver, CO, USA, 2004.

[12] P. Berman. A $d/2$ approximation for maximum weight independent set in d-claw free graphs. In *SWAT*, LNCS 1851, pages 214–219, 2000.

[13] M. Bläser and B. Manthey. Smoothed complexity theory. In *MFCS*, LNCS 7464, pages 198–209, 2012.

[14] G. Blin and R. Rizzi. Conserved interval distance computation between non-trivial genomes. In *COCOON*, LNCS 3595, pages 22–31, 2005.

[15] B. Bollobás. The chromatic number of random graphs. *Combinatorica*, 8:49–55, 1988.

[16] B. Bollobás. *Random Graphs*. Cambridge University Press, second edition, 2001.

[17] D. Bryant. The complexity of calculating exemplar distances. In D. Sankoff and J. H. Nadeau, editors, *Comparative Genomics*, volume 1 of *Computational Biology*, pages 207–211. 2000.

[18] B. Chandra and M. Halldórsson. Greedy local improvement and weighted set packing approximation. In *SODA*, pages 169–176, 1999.

[19] C. Chauve, G. Fertin, R. Rizzi, and S. Vialette. Genomes containing duplicates are hard to compare. In *ICCS*, pages 783–790, 2006.

[20] J. Chen, X. Huang, I.A. Kanj, and G. Xia. Linear FPT reductions and computational lower bounds. In *STOC*, pages 212–221, 2004.

[21] X. Chen, L. Liu, Z. Liu, and T. Jiang. On the minimum common integer partition problem. In *CIAC*, LNCS 3998, pages 236–247, 2006.

[22] X. Chen, L. Liu, Z. Liu, and T. Jiang. On the minimum common integer partition problem. *ACM Transactions on Algorithms*, 5:1–18, 2008.

[23] X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, and T. Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2:302–315, 2005.

[24] Z. Chen, R. H. Fowler, B. Fu, and B. Zhu. On the inapproximability of the exemplar conserved interval distance problem of genomes. *Journal of Combinatorial Optimization*, 15:201–221, 2008.

[25] Z. Chen, B. Fu, R. Goebel, G. Lin, W. Tong, J. Xu, B. Yang, Z. Zhao, and B. Zhu. On the approximability of the exemplar adjacency number problem for genomes with gene repetitions. *Theoretical Computer Science*, 550:59–65, 2014.

[26] Z. Chen, B. Fu, and B. Zhu. The approximability of the exemplar breakpoint distance problem. In *AAIM*, LNCS 4041, pages 291–302, 2006.

[27] Z.-Z. Chen, Y. Okamoto, and L. Wang. Improved deterministic approximation algorithms for max TSP. *Information Processing Letters*, 95:333–342, 2005.

[28] Z.-Z. Chen and L. Wang. An improved approximation algorithm for the Bandpass-2 problem. In *COCOA*, LNCS 7402, pages 188–199, 2012.

[29] H. Chitsaz, R. Backofen, and S. C. Sahinalp. biRNA: fast RNA-RNA binding sites prediction. In *WABI*, pages 25–36, 2009.

[30] H. Chitsaz, R. Salari, S. C. Sahinalp, and R. Backofen. A partition function algorithm for interacting nucleic acid strands. *Bioinformatics*, 25:365–373, 2009.

[31] K.L. Chung and P. Erdös. On the application of the Borel-Cantelli Lemma. *Transactions of the American Mathematical Society*, 72:179—186, 1952.

[32] L. Comtet. *Advanced Combinatorics: the Art of Finite and Infinite Expansions*. Springer, 1974.

[33] G. Coop, X. Wen, C. Ober, J. K. Pritchard, and M. Przeworski. High-resolution mapping of crossovers reveals extensive variation in fine-scale recombination patterns among humans. *Science*, 319:1395–1398, 2008.

[34] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[35] A. Deshpande and D. A. Spielman. Improved smoothed analysis of the shadow vertex simplex method. In *FOCS*, pages 349–356, 2005.

[36] L. Devroye. A probabilistic analysis of the height of tries and of the complexity of triesort. *Acta Informatica*, 21:229–237, 1984.

[37] R. Diestel. *Graph Theory*. Springer, third edition, 2005.

[38] R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[39] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45:634–652, 1998.

[40] P. Flajolet. On the performance evaluation of extendible hashing and trie search. *Acta Informatica*, 20:345–369, 1983.

[41] P. Flajolet and J. M. Steyaert. A branching process arising in dynamic hashing, trie searching and polynomial factorization. In *ICALP*, LNCS 140, pages 239–251, 1982.

[42] E. Fredkin. Trie memory. *Communications of the ACM*, 3:490–499, 1960.

[43] H. N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *STOC*, pages 448–456, 1983.

[44] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, first edition, 1979.

[45] O. Gascuel. *Mathematics of Evolution and Phylogeny*. Oxford University Press, 2007.

[46] M. M. Halldórsson, J. Kratochvíl, and J. A. Telle. Independent sets with domination constraints. *Discrete Applied Mathematics*, 99:39–54, 2000.

[47] M. Halldórsson. Approximating the minimum maximal independence number. *Information Processing Letters*, 46:169–172, 1993.

[48] S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM*, 46:1–27, 1999.

[49] F. Harary. *Graph Theory*. Addison-Wesley, 1969.

[50] R. Hassin and S. Rubinstein. Better approximations for max TSP. *Information Processing Letters*, 75:181–186, 2000.

[51] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *FOCS*, pages 627–636, 1996.

[52] F. W. D. Huang, J. Qin, C. M. Reidys, and P. F. Stadler. Partition function and base pairing probabilities for RNA-RNA interaction prediction. *Bioinformatics*, 25:2646–2654, 2009.

[53] L. Huang, W. Tong, R. Goebel, T. Liu, and G. Lin. A 0.5358-approximation for Bandpass-2. *Journal of Combinatorial Optimization*, pages 1–15, 2013.

[54] H. Jiang, G. Lin, W. Tong, B. Zhu, and D. Zhu. Isomorphism and similarity for 2-generation pedigrees. In *ISBRA 2014*, LNCS/LNBI 8492, pages 396–396, 2014.

[55] M. Jiang. The zero exemplar distance problem. *Journal of Computational Biology*, 18:1077–1086, 2011.

[56] V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37:27–35, 1991.

[57] H. Kaplan and N. Shafrir. The greedy algorithm for shortest superstrings. *Information Processing Letters*, 93:13–17, 2005.

[58] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[59] J. A. Kelner and D. A. Spielman. A randomized polynomial-time simplex algorithm for linear programming. In *STOC*, pages 51–60, 2006.

[60] L. G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20:53–72, 1980.

[61] D. E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley, 1973.

[62] L. Kučera. The greedy coloring is a bad probabilistic algorithm. *Journal of Algorithms*, 12:674–684, 1991.

[63] A. X. Li, M. Marz, J. Qin, and C. M. Reidys. RNA-RNA interaction prediction based on multiple sequence alignments. *Bioinformatics*, 27:456–463, 2011.

[64] G. Lin. On the Bandpass problem. *Journal of Combinatorial Optimization*, 22:71–77, 2011.

[65] T. Łuczak. The chromatic number of random graphs. *Combinatorica*, 11:45–54, 1991.

[66] Saad M. On the approximation of optimal structures for RNA-RNA interaction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6:682–688, 2009.

[67] B. Manthey and K. Plociennik. Approximating independent set in perturbed graphs. *Discrete Applied Mathematics*, 161:1761–1768, 2013.

[68] B. Manthey and H. Röglin. Smoothed analysis: analysis of algorithms beyond worst case. *it - Information Technology*, 53:280–286, 2011.

[69] H. Mendelson. Analysis of extendible hashing. *IEEE Transactions on Software Engineering*, 8:611–619, 1982.

[70] I. M. Meyer. Predicting novel RNA-RNA interactions. *Current Opinion in Structural Biology*, 18:387–393, 2008.

[71] D. L. Miller and J. F. Pekny. A staged primal-dual algorithm for perfect b-matching with edge capacities. *ORSA Journal on Computing*, 7:298–320, 1995.

[72] D. R. Morrison. Patricia—practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM*, 15:514–534, 1968.

[73] U. Mückstein, H. Tafer, J. Hackermüller, S. H. Bernhart, P. F. Stadler, and I. L. Hofacker. Thermodynamics of RNA-RNA binding. *Bioinformatics*, 22:1177–1182, 2006.

[74] M. Ng, D. Levinson, and S. Faraone *et al.* Meta-analysis of 32 genome-wide linkage studies of schizopherenia. *Mol Psychiatry*, 14:774–785, 2009.

[75] S. Ng, K. Buckingham, and C. Lee *et al.* Exome sequencing identifies the cause of a mendelian disorder. *Nature Genetics*, 42:30–35, 2010.

[76] C. T. Nguyen, Y. C. Tay, and L. Zhang. Divide-and-conquer approach for the exemplar breakpoint distance. *Bioinformatics*, 21:2171–2176, 2005.

[77] S. Nilsson and M. Tikkanen. An experimental study of compression methods for dynamic tries. *Algorithmica*, 33:19–33, 2002.

[78] K. Paluch, M. Mucha, and A. Madry. A 7/9-approximation algorithm for the maximum traveling salesman problem. In *APPROX-RANDOM*, LNCS 5687, pages 298–311, 2009.

[79] D. D. Pervouchine. Iris: intermolecular RNA interaction search. *Genome Inform*, 15:92–101, 2004.

[80] B. Pittel. Asymptotical growth of a class of random trees. *Annals of Probability*, 13:414–427, 1985.

[81] B. Pittel. Path in a random digital tree: limiting distributions. *Advances in Applied Probability*, 18:139–155, 1986.

[82] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *STOC*, pages 475–484, 1997.

[83] M. Régnier. On the average height of trees in digital searching and dynamic hashing. *Information Processing Letters*, 13:64–66, 1981.

[84] H. J. Romero, C. A. Brizuela, and A. Tchernykh. An experimental comparison of two approximation algorithms for the common superstring problem. In *ENC*, pages 27–34, 2004.

[85] R. Salari, R. Backofen, and S. C. Sahinalp. Fast prediction of RNA-RNA interaction. *Algorithms for Molecular Biology*, 5:5–5, 2010.

[86] A. Sankar. *Smoothed analysis of Gaussian elimination.* PhD thesis, Massachusetts Institute of Technology, Deptartment of Mathematics, 2004.

[87] D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15:909–917, 1999.

[88] A. I. Serdyukov. An algorithm with an estimate for the traveling salesman problem of the maximum. *Upravlyaemye Sistemy*, 25:80–86, 1984.

[89] D. A. Spielman and S.-H. Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. In *STOC*, pages 296–305, 2001.

[90] D. A. Spielman and S.-H. Teng. Smoothed analysis (motivation and discrete models). In *WADS*, LNCS 2748, pages 256–270, 2003.

[91] D. A. Spielman and S.-H. Teng. Smoothed analysis of algorithms and heuristics. In L. M. Pardo, A. Pinkus, E. Suli, and M. J. Todd, editors, *Foundations of Computational Mathematics, Santander 2005*, pages 274–342. Cambridge University Press, 2006.

[92] D. A. Spielman and S.-H. Teng. Smoothed analysis: An attempt to explain the behavior of algorithms in practice. *Communications of the ACM*, 52:76–84, 2009.

[93] A. H. Sturtevant and T. G. Dobzhansky. Inversions in the third chromosome of wild races of *drosophila pseudoobscura* and their use in the study of the history of the species. *Proceedings of National Academy of Sciences (United States of America)*, 22:448–450, 1936.

[94] J. S. Sun and J. L. Manley. A novel U2-U6 snRNA structure is necessary for mammalian mRNA splicing. *Genes & Development*, 9:843–854, 1995.

[95] W. Szpankowski. Some results on v-ary asymmetric tries. *Journal of Algorithms*, 9:224–244, 1988.

[96] W. Szpankowski. Digital data structures and order statistics. In *WADS*, LNCS 382, pages 206–217, 1989.

[97] Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22:215–225, 1975.

[98] W. Tong, Z.-Z. Chen, L. Wang, Y. Xu, J. Xu, R. Goebel, and G. Lin. An approximation algorithm for the Bandpass-2 problem. *CoRR*, abs/1307.7089, 2013.

[99] W. Tong, R. Goebel, W. Ding, and G. Lin. An improved approximation algorithm for the Bandpass problem. In *FAW-AAIM*, LNCS 7285, pages 351–358, 2012.

[100] W. Tong, R. Goebel, and G. Lin. Approximating the minimum independent dominating set in perturbed graphs. In *COCOON*, LNCS 7936, pages 257–267, 2013.

[101] W. Tong, R. Goebel, and G. Lin. Approximating the minimum independent dominating set in perturbed graphs. *Theoretical Computer Science*, 554:275–282, 2014.

[102] W. Tong, R. Goebel, and G. Lin. On the smoothed heights of Trie and Patricia index trees. In *COCOON*, pages 94–103, 2014.

[103] W. Tong, R. Goebel, T. Liu, and G. Lin. Approximation algorithms for the maximum multiple RNA interaction problem. In *COCOA*, LNCS 8287, pages 49–59, 2013.

[104] W. Tong, R. Goebel, T. Liu, and G. Lin. Approximating the maximum multiple RNA interaction problem. *Theoretical Computer Science*, 556:63–70, 2014.

[105] W. Tong and G. Lin. An improved approximation algorithm for the minimum common integer partition problem. In *ISAAC*, pages 353–364, 2014.

[106] R. Vershynin. Beyond hirsch conjecture: Walks on random polytopes and smoothed complexity of the simplex method. In *FOCS*, pages 133–142, 2006.

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

[108] C. Wang. The independent domination number of random graph. *Utilitas Mathematica*, 82:161–166, 2010.

[109] G. A. Watterson, W. J. Ewens, T. E. Hall, and A. Morgan. The chromosome inversion problem. *Journal of Theoretical Biology*, 99:1–7, 1982.

[110] B. Wieland and A. P. Godbole. On the domination number of a random graph. *The Electronic Journal of Combinatorics*, 8(1), 2001.

[111] D. P. Woodruff. Better approximations for the minimum common integer partition problem. In *APPROX-RANDOM*, LNCS 4110, pages 248–259, 2006.

[112] R. Fowler Z. Chen, B. Fu and B. Zhu. Lower bounds on the approximation of the exemplar conserved interval distance problem of genomes. In *COCOON*, LNCS 4112, pages 245–254, 2006.

[113] W. Zhao, P. Zhang, and T. Jiang. A network flow approach to the minimum common integer partition problem. *Theoretical Computuing Science*, 369:456–462, 2006.

[114] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *STOC*, pages 681–690, 2006.