UNIVERSITY OF CALIFORNIA
RIVERSIDE

Algorithms and Software for PCR Primer Design

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Yu-Ting Huang

June 2015

Dissertation Committee:

Professor Marek Chrobak, Chairperson
Professor Neal Young
Professor Tao Jiang
Professor Stefano Lonardi

The Dissertation of Yu-Ting Huang is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I sincerely thank my adviser, Marek Chrobak, who patiently guides me through the PhD program for past five years, spending time to help me finishing all works including this dissertation. Without his help, I would not have been here. I want to thank my committee, Tao Jiang, Stefano Lonardi, and Neal Young for helpful comments on my research and this dissertation.

Friendly and supportive Lab mates are very helpful when I feel stressed or need any practical help. I am glad that Yi-Wen Yang, Li Yan and other algorithm lab people are around me for discussion or relaxing, you all make my PhD experience happy and worth memory.

I also have very thoughtful house mates, who endure my absence and help me saving time while I am busy , and cheers me up when I am stressed. I am grateful to have Johann Chu and Mike Li being around.

To my parents, who always have faith on my endeavor

ABSTRACT OF THE DISSERTATION

Algorithms and Software for PCR Primer Design

by

Yu-Ting Huang

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, June 2015
Professor Marek Chrobak, Chairperson

Polymerase Chain Reaction (PCR) is a widely used technology in molecular biology for DNA amplification. To generate multiple copies of a DNA molecule, a pair of primers (two synthesized DNA sequences with a total length of 15-30 bases) are annealed to the boundaries of the targeted DNA molecule. Then, the new replicated DNA fragment elongates from one primer to the other.

Though primers always hybridize to their respective complements within DNA sequences, primer pairs for targeted DNA sequences can also anneal non-targeted DNA fragments containing common DNA sub-sequences also found in targeted DNA molecules. During the PCR process, primer pairs that offer high specificity and coverage rates for targeted fragments among all the copies are preferred.

To provide primer pairs with high selectivity, several computational algorithms have been proposed. Most state-of-the-art algorithms take into account signature primers, or common short DNA fragments in the targeted DNA molecules. However, these algorithms do not account for the fact that during the PCR process in which primer pairs designed using signature primers are used, DNA fragments that do not have signature primers will not become amplified. These algorithms are, then, limited in various ways.

Predicting primers' respective binding affinities is crucial in primer design because, during the PCR process, the annealing between the targeted DNA fragments and the primers with low binding affinity degenerates during the PCR process's thermal cycles. Because of this degeneration, targeted fragments expected to be reproduced by the primer pairs go missing during DNA amplification.

It is important to note that a particular primer's nucleic acids do not contribute equally to the binding affinity. Specifically, this binding affinity is determined by the nucleic acids in the 3' end of the primer more than the nucleic acids in the 5' end. Existing algorithms typically oversimplify their predictions by either ignoring primers with high binding affinity or including primers with low binding affinity.

To address current algorithms' limitations, we created PRISE2, a robust computational tool for sequence-selective PCR primer design. This innovative tool considers all subsequences of potential primer pairs to increase the coverage rate of the targeted fragments. This tool also provides a flexible mechanism with which to formulate positional bias when estimating primers' binding affinity. Importantly, the execution time of locating binding sites for all potential primers is positively proportion to the number of the subsequences. To accelerate searching for the binding sites, this tool clusters subsequences according to their sequence prefices to reduce the searching space. PRISE2 not only provides a user-friendly interface, but also offers full functionality for primer-design tasks. It was implemented using C++ and Qt frameworks to guarantee efficiency and achieve a cross-platform requirement.

In applications where a collection of similar sequences need to be amplified using PCR, *degenerate primers* can be used to improve the efficiency and accuracy of amplification, since they can hybridize into multiple, unique DNA fragments. Conceptually, degenerate primers allow multiple bases at various positions. However, in reality, they are mixtures of regular primers that differ on certain bases. Specific degenerate primers' *degeneracy* refers to the number of regular primers in a mixture. Higher degeneracy allows a primer to amplify more targeted sequences simultaneously and also leads to low specificity for targeted sequences that adversely affect the quality and quantity of amplification. It is essential to find a good balance between high coverage and low degeneracy, a balance that a tool like PRISE2 helps achieve.

For degenerate primer design, we proposed a new heuristic algorithm, $RRD2P$, to compute degenerate primer pairs with near-optimal coverage to targets under the specified degeneracy threshold. $RRD2P$ runs in polynomial time and is confirmed to produce primer pairs with good coverage on three biological data sets. This production compares favorably with a similar tool called HYDEN. The fundamental goal driving $RRD2P$: to represent computing optimal primers as an integer linear program, solve their fractional relaxation, and then apply randomized rounding to obtain an integral solution.

Contents

List of Figures

Part I

Primer Design Problem

Chapter 1

Introduction

1.1 PCR and Primers

Polymerase Chain Reaction (PCR) is an amplification technique widely used in molecular biology to generate multiple copies of a desired region of a given DNA sequence. PCR process uses two small pieces of synthetic DNA sequences called primers, which are typically of length 15-30 bases. This process requires the primer pair to identify the boundary of amplification. This pair of primers, comprised of what are known as forward and reverse primers, can be obtained from 5' ends of each individual strand in a target sequence. These ends and their primers are complementary to the 3' ends of another strand.

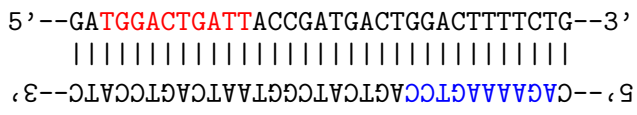
Generally speaking, a PCR procedure entails a series of cycles. These cycles consist of following 3 phases:

1. *Denaturation:* The denaturation step entails using a double-stranded DNA target sequence. In this step, the reaction is heated to cause the DNA to melt, yielding two single-stranded DNA molecules.

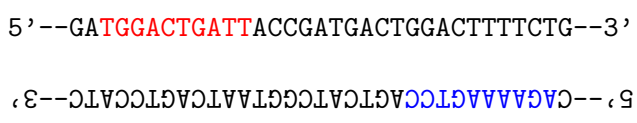
2. *Annealing:* In the annealing step, the temperature is lowered and two primers anneal to each of the single-stranded DNA templates. Stable DNA-DNA hydrogen bounds are formed between primers and templates.

3. *Extension:* In the extension step, starting from the primer binding sites, the DNA polymerase synthesizes a new DNA strand complementary to the template strand toward 5' end.

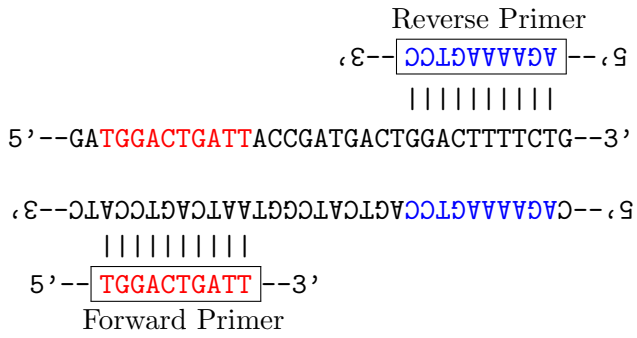
Each cycle doubles the number of the desired regions in the target sequence. The PCR procedure repeats this cycle until obtaining enough duplication (see Figure 1.1). If a primer is able to hybridize to a sequence and continue the PCR process, we say that it "covers" this sequence.



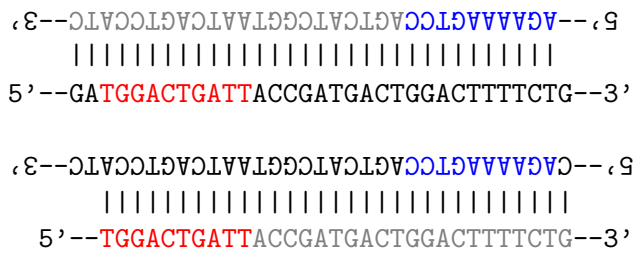
(a) Target sequence



(b) Denaturation



(c) Annealing



(d) Extension

Figure 1.1: PCR cycle

In fields like genetics, environmental sciences, and medicine, selective PCR is

widely used, a process in which amplification selectivity is required. For example, in applications like single-nucleotide polymorphism (SNP) analyses [23, 43, 45, 54] or monitoring of environmental microorganism populations [33, 39, 47], both the isolation and amplification of specific DNA fragments - referred to as target sequences - from environmental samples is essential. These samples are often complex mixtures of DNA fragments that may also include non-target sequences, which are pieces of DNA similar to target sequences that represent genes or organisms that are not the subject of the study. Naturally, such non-target sequences should not appear in the amplified product. A critical step in achieving such high-resolution amplification is the design of sequence-selective PCR primers, or primers that amplify only the target sequences.

Designing such PCR primers manually is tedious and time consuming. Importantly, though, this process can be both streamlined and greatly simplified with the aid of appropriate software. Such software tools can be used to accomplish following - as well as various other - key tasks:

1. Identify target and non-target sequences or quickly evaluate the quality of a large number of primer candidates by computing their alignment with these sequences.

2. Compute primers' biological parameters, such as GC-content or melting temperature, which could be essential for evaluating whether or not primers will produce specific amplification and high yield.

3. Identify primers with appropriate complementarity properties, preventing undesirable PCR side effects where primers bind to themselves by forming hairpin or primer-dimer structures.

1.2 Degenerate Primers

In some applications that typically involve many very similar but not identical sequences, such as studying the composition of microbial communities or analyzing specific gene in different organisms, a modification of PCR called *Multiplex PCR (MP-PCR)* [19] is usually applied. This modification of PCR amplifies multiple target sequences simultaneously in single procedure by using multiple primers along with temperature-mediated DNA polymerase. In such applications, *degenerate primers* can be used to improve the efficiency and accuracy of amplification.

Degenerate primers [37] can be thought of, on a conceptual level, as having *ambiguous bases* at certain positions, that is, bases that represent several different nucleotides. This ambiguity enables degenerate primers to bind to several different sequences, thus allowing for the amplification of multiple sequences during a single PCR experiment. Degenerate primers are represented as strings formed from IUPAC codes. Each code represents multiple possible alternatives for each position in a primer sequence (see Table 1.1). In reality, degenerate primers are simply appropriate mixtures of regular primers that are as cheap and easy-to-produce as regular primers. This simplicity and low cost makes them widely used in MP-PCR.

IUPAC nucleotide code	M	R	W	S	Y	K	V	H	D	B	N
represented bases	A C	A G	A T	 C G	 C T	 G T	A C G	A C T	A G T	 C G T	A C G T

Table 1.1: IUPAC nucleotide code table for ambiguous bases.

Given a target template `GATGGACTGATTACCGATGACTGGACTTTTCTG`, the degenerate primer `TRGASTGATY` matches the substring `TGGACTGATT` of the target sequence perfectly, since bases `R`, `S`, and `Y` represent bases `G`, `C`, and `Y` respectively. The *degeneracy* $\deg(p)$ of a primer p equals the number of distinct non-degenerate primers it represents. For

example, the degeneracy of primer $p =$ `ACMCM` is 4 because it represents the following four non-degenerate primers: `ACACA`, `ACACC`, `ACCCA`, and `ACCCC`.

Although primers with higher degeneracy can cover more target sequences, high degeneracy can also negatively impact the quality and quantity of amplification. For example, including too many primers in the mixture could lead to problems like mis-priming (i.e., unrelated sequences may be amplified) or primer cross-hybridization (i.e., primers may hybridize to each other). Thus, when designing degenerate primers, it is essential to find a good balance between high coverage and low degeneracy.

1.3 Contribution

Depending on the application, primer design may involve designing either regular (non-degenerate) primers or degenerate primers. In this thesis, and as stated below, we contribute to both problems:

Regular Primer Design. Regular primer design is simpler since we do not need to consider the trade-off between the degeneracy and other constraints; it essentially reduces to finding most common substrings among target sequences.

For regular primer design, we created a robust tool called PRISE2 that pipelines and speeds up the primer design process. PRISE2 aims to find sequence-selective PCR primers and probes. To achieve high level of selectivity, PRISE2 users can specify target sequences that are supposed to be amplified as well as non-target sequences that should be avoided. The program emphasizes primer selectivity on the 3' end, which is crucial for the selective amplification of conserved sequences, such as rRNA genes.

PRISE2 offers users a graphical wizard that guides users as they specifying their parameters for hybridization and the desired properties of primers, including length and GC content. After obtaining a list of possible primer pairs using the user-friendly interface, users can further interactively manipulate the list of parameters and choose primer pairs that are best suited for their needs. More advanced features include, for example, the capability to define a custom mismatch penalty function.

Degenerate Primer Design. We also worked on the degenerate primer design tasks. Degenerate primer design is more complicated than regular primer design. There are several

ways to formalize the problem of degenerate primer design in the language of combinatorial optimization.

In this paper, we focus on the *Maximum Coverage Degenerate Primer Design* problem (**MCDPD**) in which we optimize the coverage of the primer. **MCDPD** has been proven to be NP-hard and we are interested in designing approximation algorithms for **MCDPD** and establishing lower bounds on its approximation ratio.

MCDPD can be formulated in terms of computing bi-cliques in certain bipartite graphs. Motivated by this application, in addition to **MCDPD**, we studied several other versions of the maximum bi-clique problem. Specifically, we introduced and studied problems that we called **ThrPairBiClique**, **ThrBiClique**, and **MaxBiClique**.

MaxBiClique can be solved in polynomial time, either by the reduction to matching or using total unimodularity. However, **MCDPD**, **ThrPairBiClique**, and **ThrBiClique** are NP-hard. We show that the natural integer linear programs for these three problems have large integrality gaps. Using the PCP theorem, we prove that **ThrPairBiClique** does not have an n^{ϵ}-approximation algorithm, unless P=NP. We also articulate some relations between approximation ratios of these three problems.

Our main objective is to establish tight upper or lower bounds on the approximation ratio for **MCDPD**. Although, as of now, we have not been able to accomplish this objective, studying other versions of the maximum bi-clique problem will shed some light on the computational complexity of **MCDPD**.

MCDPD has been proven to be NP-hard; thus there is no polynomial time algorithm to solve this problem. However, much effort has been put forth to develop heuristic algorithms. For example, degenerate primer design can be accomplished by iteratively finding some primers with high coverage. In this thesis, we propose a new heuristic algorithm, $RRD2P$, to compute degenerate primer pairs with near-optimal coverage to targets under the specified degeneracy threshold. $RRD2P$ runs in polynomial time and is confirmed to produce primer pairs with good coverage on three biological data sets. This algorithm produces results that compare favorably with a similar tool called HYDEN. The fundamental idea behind $RRD2P$ is to represent computing optimal primers as an integer linear program, solve their fractional relaxation, and then apply randomized rounding to obtain an integral solution.

The above randomized rounding algorithm represents **MCDPD** as a linear program. Computing a fractional solution of this linear program is the most time consuming

module in $RRD2P$. In an effort to speed up this process, we propose a polynomial-time combinatorial algorithm for solving this linear program, for the restricted case of **MCDPD** with binary alphabet and no mismatches.

Part II

Automation Tools for Primer Design

Chapter 2

Tool for Primer/Probe Design

2.1 PRISE2

To help in the primer design process, we created a new software package for designing sequence-selective PCR primers and probes called PRISE2 (PRImer SElector 2). PRISE2 is the successor of an existing tool, PRISE [25]. PRISE2 preserves PRISE's main design process strategy and core algorithms. PRISE2 also provides a number of extensions and new features, including platform independence, performance enhancements, and the ability to add probes to selected primer pairs,.

In a nutshell, PRISE2 is an interactive software package for designing primers and probes for PCR experiments that accepts sets of target and non-target sequences on input. It produces primer pairs that will typically selectively amplify the target sequences. PRISE2 includes customizable features and settings to ensure that computed primers will be useful during various applications.

One distinctive feature of PRISE2 is that it allows users to emphasize primer selectivity at the 3' end. This feature is essential for selective amplification of conserved sequences, such as rRNA genes. This features also allows users to accurately differentiate between target and non-target sequences during the actual laboratory PCR process [38, 29, 11]. The need for such precise selectivity arises when amplifying sequences from mixtures of DNA. Examples of such applications include subtype analyses, in which very similar groups of genes need to be distinguished, or rRNA gene studies that monitor population levels of specific microorganisms in environmental samples containing millions of different organisms [33, 39, 47]. In fact, PRISE2 - as well as its predecessor, PRISE - was specifically designed

to support studies of microbial communities [57, 56, 46, 17, 60, 16]. These studies often involve analyzing environmental samples containing mixtures of both target and non-target sequences.

Another application where such high selectivity is useful: genomic walking [11, 44, 5, 10]. During this application, the focus on the 3' end selectivity is implemented through a detailed and customizable per-position mismatch allowance matrix that sets more stringent mismatch criteria at the 3' end than for the rest of the primer. We are not aware of any other software that provides such sophisticated selectivity settings.

When searching for candidate primers, PRISE2 uses a custom algorithm that assigns appropriate mismatch weights to different positions when aligning candidate primers against their potential binding regions.

Other programs typically use local alignment tools, such as BLAST, to do similarity searches used to identify the binding position of primers. Such searches may result in the loss of some match information over the entire primer range, especially when the match is not perfect at the primer ends.

Another distinctive feature of PRISE2 is the probe design function, which is useful during quantitative analyses to measure the amplification of target sequences. For example, this function enables the design of Taqman and other primer-probe based assays as well as probes for FISH and other hybridization-based assays. The probe design feature is integrated with the primer design process in the sense that users can add probes to selected primer pairs. This feature allows for triples consisting of a forward primer, reverse primer, and a probe to be evaluated in tandem in terms of coverage and other quality criteria.

When designing probes, users can specify biological parameters that may be different than those for primers. For example, setting the melting temperature ranges for probes and primers at different levels helps to ensure proper functioning of probes during the Taqman PCR process. The mechanism for defining mismatch criteria for probes is different than that used for primers and mismatches near the middle of the probe given more weight than at the ends. For this reason, the algorithm for selecting probes is also quite different than that used for primers.

Unlike some other programs [9, 12, 24, 34, 26], PRISE2 does not require signature primers, or short pieces of DNA that are only conserved in target sequences. Using signature primers reduces computational cost. However, not all groups have unique signatures [26]. PRISE2 considers all combinations of forward/reverse primers and probes so that, for such

groups, it can find individually non-specific but group-specific primer sets, for which the methods using signature primers are not likely to be effective.

PRISE2 also differs from existing tools in terms of its fundamental approach to the design process - a process that emphasizes interaction with the user to take advantage of human expertise. Rather than simply computing a primer-probe set that optimizes user-specified criteria, at several stages of the process, PRISE2 produces a list of candidate primers or probes. For each of these candidate primers or probes, a list of quality indicators - like coverage, length, GC-content - is generated. The user can then manipulate these lists, sorting them according to different criteria to choose a subset of candidates. This interactive approach also allows users to backtrack through the process to fine-tune the program's parameters. For example, if an insufficient number of primer candidates are produced due to excessively stringent biological settings, users can go back through previous steps and experiment with different parameters that may increase the number of candidate primers. More extensive comparison between PRISE2 and other primer design tools can be found in the "Comparison" section later in this paper.

PRISE2 is written in C++ with the graphical user interface implemented with the Qt toolkit. It is free for non-commercial usage and can be downloaded from http://alglab1.cs.ucr.edu/OFRG/PRISE2.php directly or from the OFRG website, http://algorithms.cs.ucr.edu/OFRG (users must simply follow the link to PRISE2). The software can run on the following systems and is distributed as an executable file that can be executed once downloaded (no additional installation is required):

- Windows 2000/NT/XP/7/8
- Ubuntu 9.04 or higher
- Mac OS 10.5 or higher

Activated Internet connectivity is needed for full functionality to connect to the NCBI website. For users who have target and non-target sequences already selected - or, those who have a local installation of BLAST - no Internet connection is required. PRISE2 requires a minimum of 512 MB of RAM (1 GB of RAM or more is recommended).

2.2 Comparison

In this section, we compare PRISE2 with four existing tools for primer design to emphasize PRISE2's unique characteristics.

PRISE When compared to its predecessor, PRISE, PRISE2's most significant new feature is its probe design option, which can be useful for Taqman and other primer-probe assays. In quantitative analyses like Taqman assays, probes can be used to measure the amplification of target sequences along with a pair of primers. Probes can also be designed for FISH and other hybridization-based assays.

Unlike PRISE, which is only available on Windows, PRISE2 is a cross-platform software, not restricted to a specific operating system. We created PRISE2 installations for Windows, Macintosh and Linux machines. Another new PRISE2 feature: the option to communicate with a local BLAST application, which provides a more convenient way to identify target and non-target sequences without the need for connecting to the NCBI website.

Although PRISE2's core algorithms for sequence similarity and identifying primer candidates remain the same as those in the original PRISE, several performance improvements were made to speed up tool's computation abilities for large data sets. For instance, the module for computing primer candidates is roughly 4 times faster in PRISE2 than in PRISE. When run on a data set of 123 target and 319 non-target sequences from the NCBI flu database (each of length approximately 1000bps, with most parameters at default values), PRISE2 completed the task in 12 minutes. When confronted with the same task, PRISE clocked in at 48 minutes on a Windows 8 machine with 8GB memory and 2.4GHz CPU. PRISE2 completed the same task, in other words, 36 minutes faster than PRISE.

PrimerProspector One other tool that allows users to differentiate selectivity properties for the 3' and 5' ends is PrimerProspector [53]. This tool allows different weights to be applied to the 3' and 5' ends of primers to estimate the likelihood of binding. To consider a primer candidate as functional to a specific template, PrimerProspector requires this candidate to have a user-specified number of continuous base matches starting at the 3' end.

This requirement feature can be thought of as a restricted case of PRISE2's 3' end selectivity mechanism. PRISE2 gives the user more control than PrimerProspector, then, to cover scenarios in which primer-template binding is likely to occur even in the absence of continuous matches at the 3' end. Additionally, unlike PRISE2, PrimerProspector does not use non-target sequences. PRISE2, however, allows users to interactively design probes for selected primer pairs.

PRIMER3 Primer3 [51] is a widely used web-based program for primer design that provides similar options to those in PRISE2. In Primer3, users can design a primer set consisting of a forward primer, a reverse primer, and a probe. Much like in PRISE2, they can also choose a number of settings for controlling the quality of the primers.

Primer3 focuses, however, on designing primer sets targeting single sequences and it does not take into consideration non-target sequences. This program also does not allow mismatches to be placed on the 3' ends of the primers. In Primer3, the design emphasizes a single-shot process without the flexibility and convenience of the interactive procedure available in PRISE2.

Primer-BLAST Primer-BLAST [59] is an integrated tool for primer design provided by NCBI. This tool combines Primer3 and BLAST functionalities along with the Needleman-Wunsch (NW) global alignment algorithm - an algorithm that overcomes the shortcomings of local alignment for primer design purpose.

Primer-BLAST first utilizes Primer3 to generate the candidate primer pairs, then applies the specific BLAST to check for specificity. Since the program consists of two separate modules, users can check the specificity of existing primer candidates. However, Primer-BLAST inherits Primer3's limitations, as it does not have the feature to design a desired set of non-target sequences and has limited abilities to control mismatched positions. Primer-BLAST achieves identifying primers' specificity in a specific region at 3' end by forcing the number of mismatches between primers and unintended targets from a user-specified database.

QuantPrime and PRIMEGENS QuantPrime [8] and PRIMEGENS [55, 50] are two other software packages used for primer design. Neither of these tools include options to specify non-target sequences. They also do not allow for users to design probes. For the 3' end selectivity, QuantPrime provides only a rudimentary mechanism of limiting the number of mismatches near the 3' end, a feature that PRIMEGENS does not implement.

2.3 Workflow

As explained in chapter 2.1, the primer design process in PRISE2 is essentially the same as in its predecessor, PRISE. The module for probe design, however, is new. We describe the complete process in this section.

Designing primers and probes in PRISE2 is accomplished in three stages:

1. *Identification of target and non-target sequences.* Here, the user can download a collection of sequences from GenBank and use the provided interactive tools to choose from them - or, from other collections of sequences - the desired sets of target and non-target sequences.

2. *Generation of candidate primer pairs.* In this step, the program computes a set of primer candidates, according to the specified parameters, and groups them into primer

pairs. Then the user can use a variety of sorting tools to manipulate the list of these primer pairs to choose a smaller collection of high quality pairings.

3. *Adding probes.* Once these desired primer pairs have been selected, the next module of PRISE2 allows probes to be added to each primer pair. For each primer pair, PRISE2 determines a list of candidate probes. This module produces a collection of primer-probe sets that can be sorted according to multiple criteria, ultimately allowing the user to choose the final collection of primer-probe sets best suited for their individual PCR experiments.

Below we give a detailed description of these steps.

2.3.1 Identifying Target / Non-target Sequences

In this step, users first need to identify the seed sequences and create the hit table. The seed sequences are DNA sequences that the user aims to amplify. The hit table is a file created by subjecting the seed sequences to an analysis using BLAST (blastn) [4]. This file contains summary information (e.g., species name, etc.) for similar or related sequences from all species or organisms in specified databases. Users can either run BLAST remotely through the NCBI website (following the instructions provided in PRISE2) or locally (if they have an installation of BLAST on their machine). In the latter case, PRISE2 provides an interface to integrate sequence identification with a locally installed BLAST application and database.

Once the seed sequences and the hit table are identified and saved, PRISE2 will download associated records from GenBank and perform pairwise similarity (percentage identity) analyses of sequences. These sequences are then displayed as a list, and can be interactively manipulated and sorted according to specified properties, similarity to seed sequences, or various other attributes. This sorting feature allows users to identify the desired target and non-target sequences, which then can be saved for future use. In a typical application, the sequences most similar to seed sequences can be designated as target sequences, while non-target sequences can be selected from the remaining set. Other sequence attributes provided by PRISE2, like sequence length, may be useful in this task.

2.3.2 Designing Primers

To design primers, users first need to specify the sets of target and non-target sequences. In most situations, these sequences are those identified in the previous step, but this is not a requirement. Some users may wish to use their own collections of target or non-target sequences - sequences obtained earlier from either PRISE2 itself or from some other source. PRISE2 will accept any collections of sequences as long as they are in the FASTA format.

Next, users can specify desired properties of both individual primers and pairs of primers. These properties include, for example, primer length, GC content, and melting temperature. Among other options, users can also restrict the complementarity properties of primers to assure that chosen primer candidates do not bind to themselves and that, in selected primer pairs, the forward primer does not bind to the reverse primer.

For convenience, all parameters have pre-tested default choices that are likely to work well in most typical applications. The following step examines how to navigate the primer selectivity settings. The purpose of the selectivity settings is to identify highly selective primers, namely those that will bind both to most target sequences and to as few as possible non-target sequences. These settings allow users to define what constitutes a match between a primer and a sequence and can be controlled separately for target and non-target sequences. Put simply, stringent settings correspond with near-perfect matches, while more flexible settings correspond with inexact matches. In essence, and worth noting here: more stringent settings produce fewer primer candidates than relaxed settings. As in the previous steps, typical users will use the default settings only. However, PRISE2 also allows users to customize these parameters, either through a basic interface in which they choose stringency levels or with more advanced options in which they can manually adjust the values of all parameters.

There are two categories of selectivity settings: *mismatch allowance mechanism* and *mismatch cost matrix*. The purpose of this mismatch-allowance mechanism is to emphasize primer selectivity on the 3' end, which is crucial for functional primers, as explained earlier. This allowance can be accomplished by setting limits on the accumulated number of mismatches, starting at the 3' end and ending at any position. For example, one can specify 0 mismatches on the first 3 positions, at most 1 mismatch on the first 5 positions, and at most 2 mismatches on the first 7 positions.

The mismatch cost matrix is a nucleotide-to-nucleotide dissimilarity function that reflects the likelihood of binding to occur, with smaller values representing higher likelihood. For example, in its simplest form, this matrix's entries could be 0 for equal bases (matches) and 1 for different bases (mismatches). Users can adjust these values to distinguish different types of mismatches. For example A/G and A/C mismatches may be assigned different costs.

Consider a primer candidate p and a template string t (that is, t is either a target or a non-target sequence). For a fixed location l in t, let t denote the substring of t, of the same length as p, starting at location l. For all locations in p, and using the mismatch cost matrix, PRISE2 computes the accumulated costs of mismatches, starting from the 3' end of p. If any of these values exceeds the corresponding mismatch allowance, the program determines that there is no match at location l. Otherwise, it considers position l as a potential match and it stores the total accumulated mismatch cost as the cost of aligning p and t' at location l. After computing such alignment costs for all potential match locations l, PRISE2 then chooses the most likely location in t for this primer p to bind simply by choosing the location with minimum cost. This computation is accomplished in PRISE2 with a custom string similarity algorithm specially designed for this purpose. In cases when no potential match locations in t are found, PRISE2 determines that p does not hybridize to t. PRISE2 has also an option to allow gaps in alignments. This option is implemented by assigning costs to gaps and putting a limit on the number of gaps. In the computation described above, these costs are included as costs of either nucleotide insertions or deletions. By repeating the above process for all primer candidates p and target/non-target sequences t, PRISE2 will determine which primers bind to which sequences and at which positions. At this point in this process, PRISE2 will match all primer candidates into pairs. These pairs are then presented to the user, sorted according to their selectivity measure, that is computed as follows: consider a pair f, r of primers. Let $cov_t(f, r)$ denote the coverage of this primer pair in target sequences, and let $cov_n(f)$ and $cov_n(r)$ denote the coverage of f and r, respectively, in non-target sequences. These values are normalized so that they are in the range [0,1]. The selectivity of this primer pair is then computed as below:

$$1 - (1 - cov_t(f, r))^2 - (\frac{1}{2} cov_n(f))^2 - (\frac{1}{2} cov_n(r))^2$$

Note that this computation produces high values for pairs f, r that jointly match

many target sequences, but where each of f and r, individually, matches few non-target sequences. It is especially important to emphasize here that this value is used only to sort the primer pairs in the display window so as to give the user a rough estimate of the primer pairs' quality.

The number of candidate primer pairs depends on the stringency of all settings. If too few or too many pairs are produced, PRISE2 allows the user to backtrack and adjust these settings to re-compute new candidate pairs.

After users are satisfied with the list of candidate pairs, they can review their quality criteria. Next, they can either select a small number of primer pairs to be used in an experiment or proceed to the probe design stage. To assist users with this review task, the list of primer pairs is displayed in a tabular format that shows all quality indicators for each. This format indicates the percentages of target and non-target sequences that the primers are predicted to bind to, the product length, GC-content, complementarity properties, as well as other pieces of information that can help users make decisions. For each primer pair, users can even display and examine the alignments between this pair and all target or non-target sequences. An easy-to-use graphical interface allows users to manipulate this list in an interactive fashion. Using this interface, users can remove primer pairs, manually add new primer pairs, and sort the list according to a wide variety of criteria.

2.3.3 Designing Probes

Once the user identifies and selects a collection of primer pairs of interest, PRISE2 provides an option to add probes to these pairs. This module can be also used for designing probes for FISH and other hybridization-based assays. In quantitative PCR (qPCR) analyses like Taqman assays, probes play an essential role in monitoring the amount of sequence amplification.

The process of probe design is similar to that for primers; it involves selecting probe properties, such as the probe length and GC content. One probe attribute that assures that the probes will function properly during the qPCR process is the difference of melting temperatures between the primer pair and its associated probe. For example, for Taqman assays, setting the melting temperature for probes higher than for primers ensures that the probes will bind to the target before the primers. This sequential binding is essential for consistent quantification in such assays because measurement of amplification events occurs

when DNA synthesis from the primer leads to the polymerase enzyme contacting and then cleaving the quencher located on the end of the probe. This contacting and cleaving results in a detectable fluorescent signal.

As in the primer design process, users can specify their own scoring scheme for probes, the cost function, and the mismatch allowance matrix to obtain the desired selectivity for target sequences. Unlike for primers, where matches on the 3' end are considered more important, in the case of probes, PRISE2 focuses on the middle section of the probe by allowing the user to specify the number of continuous matches near its center. Such continuous matches increase the likelihood of binding, even when some mismatches occur near the ends of the probe. These mismatch criteria can be specified separately for target and non-target sequences.

Once probe properties and selectivity settings are defined, PRISE2 will compute, for each pair of selected primers, the list of probes that match all the criteria. The final result is a list of primer-probe sets in which each of these sets consist of a pair of primers and a probe that meet all criteria. PRISE2 displays the results for this specific computation in a window with separate tabs for each primer pair that is sorted according to the selectivity function:

$$1 - (1 - cov_t(f, r, p))^2 - (cov_n(f, r, p))^2 - \left(\frac{1}{2}\big(1 - cov_t(p)\big)\right)^2 - \left(\frac{1}{2} cov_n(p)\right)^2$$

In this formula, f, r, and p denote the forward primer, the reverse primer, and the probe in a probe set. $cov_t()$ and $cov_n()$ represent the normalized coverage values in target and non-target sequences, respectively. For each primer pair, the program creates a tabulated list with rows corresponding to probes and columns, showing properties of the corresponding primer-probe set. As for primers, these lists can be manipulated and sorted by the user, according to a number of different criteria. Ultimately, by using these tools, users can determine a small number of primer-probe sets to be used in specific PCR experiments.

A more detailed description of the primer/probe design process can be found in the PRISE2 manual and tutorial provided on PRISE2's web page.

2.4 Experiments

To demonstrate that PRISE2 finds highly selective, functional primer-probe sets, we used this program to design two Taqman real-time qPCR assays, which were subsequently tested in a laboratory experiment. Primer and probe sets were designed to differentiate the rRNA ITS region of two closely related strains of the fungus *Dactylella oviparasitica* (DO50 and DO60). Sequence-selective primers and probes were designed using PRISE2 with the default settings.

qPCR was performed using a Bio-Rad iCycler MyiQTM Real-Time Detection System (Bio-Rad Laboratories, Hercules, CA, USA). The selective primers for the DO50 and DO60 assays are DO-50 F1 (`ATCGGCCTCACAAA`) and DO-50R1 (`TAACCAATTCCTTGTTGTT`) as well as DO-60 F2 (`AGCGAAACCCTCTCA`) and DO-60R2 (`TACGAGTTGTCGCAATAC`), respectively. The selective probes for the DO50 and DO60 assays are as follows:
DO-50Probe-1, [6-FAM]`AACAGCACAGTGGACCTGCC`[BHQ1a-6FAM] and
DO-60Probe-2, [6-FAM]`AAAGCTAGCGGGCACAGGC`[BHQ1a-6FAM] , respectively, where BHQ1a is Black Hole Quencher 1 (Eurofins MWG Operon, Huntsville, AL, USA). The targets are fragments of the ITS rRNA gene with sizes of 94-bp and 75-bp for DO50 and DO60, respectively.

The non-target sequence used for the design of the DO50 assay was DO60, while the non-target sequence used for the DO60 assay was DO50. The assay for DO50 produced a robust amplification signal from DO50 DNA, but no signal from DO60 DNA. A similar, but opposite, result was obtained for the DO60 assay. These results demonstrate the ability of PRISE2 to create useful primers and probes for sequence-selective assays as in Figure 2.1.

Primers designed with PRISE and PRISE2 have been used in multiple investigations of microbial community composition studies. These studies typically require amplifying targeted sequences from DNA mixtures extracted from environmental samples containing hundreds to thousands of non-target sequences. The high resolution of the primers produced by PRISE and PRISE2 tools that have the abilities to distinguish between target and non-target sequences played critical role in these studies. For example, in a study examining the role of microorganisms in inflammatory bowel disease, the use of PRISE enabled measurements of specific bacterial rRNA gene sequences in a habitat harboring hundreds or thousands of different microorganisms. These measurements provided putative

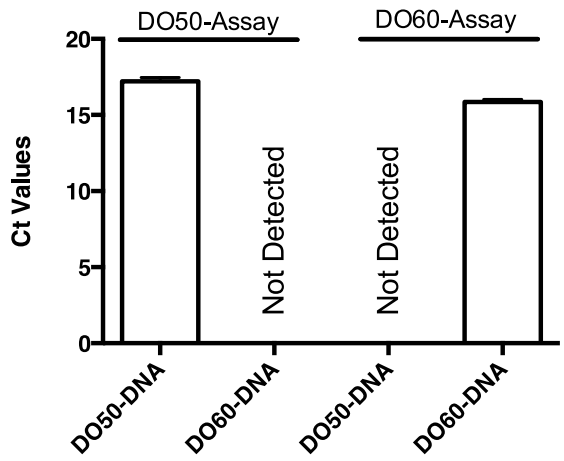


Figure 2.1: Experimental results for DO50 and DO60 assays

links between specific host-microbe interactions and disease causation.

Part III

Degenerate Primer Design

Chapter 3

Overview of Degenerate Primer Design Problems

3.1 Notations and Problem Definitions

In this chapter, we introduce notations to represent target sequences and primers, then give formal definitions to primer design problems with these notations.

Let Σ denote the alphabet of input target sequences and, in the case of DNA sequences, $\Sigma = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$. A degenerate primer is a string composed of subsets of Σ. Let $p = p_1 p_2 \dots p_k$ be a degenerate primer of length k and p_j be the j_{th} position in p, $p_j \subseteq \Sigma$ and $p_j \neq \emptyset$ for each position j. The degeneracy of p, represented as $deg(p)$, is calculated by multiplying the number of possible bases at each position (i.e., $deg(p) = \prod_{j=1}^{k} |p_j|$ where $|p_j|$ is the size of the set p_j). For example, if $p = \texttt{ACMCM}$ where $\texttt{M} = \{\texttt{AC}\}$, $deg(p) = 2 \times 2 = 4$.

Given two degenerate primers p and q with the same length k, they can be merged into a new degenerate primer $p' = p \cup q$ of length k, where $p'_i = p_i \cup q_i$ for all $i = 1, \dots, k$. For example, if $p = \texttt{AGATT[CT]}$ and $q = \texttt{AG[AC]CT[CG]}$, then $p' = p \cup q = \texttt{AG[AC][CT]T[CGT]}$. Merging will increases the degeneracy (i.e., $deg(p') \geq deg(p)$ and $deg(p') \geq deg(q)$). However, $deg(p')$ is not a function of $deg(p)$ and $deg(q)$, as the increase of degeneracy depends on the number of degenerate positions, the number of bases at each position, and the relationship between them.

We say that a degenerate primer p *covers* a target sequences s if at least one of the regular primer represented by p occurs in s as a substring, so that p can hybridize to

the reverse-complementary strand of s. Formally, we say that a primer p of length k covers a string s of length l if there exists a position r such that $s_{r+j} \in p_j$ for all $j = 1 \cdots k$. In practice, a primer can still hybridize to target sequences and be functional even if it only approximately matches the template. We will say that p covers s with at most λ mismatches if there exists a substring s' of s of length $|p|$, such that p matches s' on at least $|p| - \lambda$ positions. We refer to λ as *mismatch allowance* since we allow up to λ mismatches and still consider as p covers s. The coverage of p, denoted by $cov(p)$, is the number of target sequences that p covers.

From noted earlier, primer design can be optimized toward 3 criteria: (1) high coverage, (2) low degeneracy, and (3) small number of primers. Here, we focus on maximizing the coverage, restricting the number of primers to be 1, and binding the degeneracy with a specified threshold d. Generally speaking, we aim to find a primer p that covers the maximum number of target sequences and with the degeneracy $deg(p) \leq d$.

The *Degenerate Primer Design* problem (**DPD**) is defined in Problem 1.

Problem 1 (DPD) *Given a set of n target strings $S = \{a^1, a^2, \ldots, a^n\}$ over alphabet Σ of arbitrary lengths, integers m and d, find a degenerate primer p of length m and degeneracy at most d that covers the maximum number of strings in S.*

Finding a maximum-coverage degenerate primer is a variant of the degenerate primer design problem in which we fix the degeneracy to maximize the coverage. The **MDPD** problem can be solved by finding each primer with maximum coverage, one at a time, until all sequences are covered. Thus, the maximum-coverage DPD problem is extremely useful in practical primer design.

Intuitively, a primer will bind to conserved regions among target sequences and these regions can be identified in advance as expected primer binding sites. Therefore, we can design primers based only on these regions and assume that all input sequences are of the same length m when designing degenerate primers of length m for simplification.

The corresponding problem, maximum coverage degenerate primer design(**MCDPD**), is defined as in Problem 2.

Problem 2 (MCDPD) *Given a set of n target strings* $A = \{a^1, a^2, \ldots, a^n\}$ *over alphabet* Σ *each of length* m*, and an integer* d*, find a degenerate primer* p *of length* m *and degeneracy at most* d *that covers the maximum number of strings in* A*.*

The following is an instance of **MCDPD** over binary alphabet:
Given $A = \{$`11000, 10100,10010, 01010`$\}$ and the degeneracy threshold $d = 8$, the best degenerate primer will be `NNONO` that covers a^1, a^3 and a^4, where `N` denotes the only ambiguous base $\{0, 1\}$.

As mentioned previously, primers can be functional with a small number of mismatches in the template. Thus, the **MCDPD** with mismatch (**MCDPD**^{mis}) problem is a variant of **MCDPD**, which takes mismatch allowance into consideration as in Problem 3.

Problem 3 (MCDPD^{mis}) *Given a set of n target strings* $A = \{a^1, a^2, \ldots, a^n\}$ *over alphabet* Σ *each of length* m*, integers* d *and* λ*, find a degenerate primer* p *of length* m *and degeneracy at most* d *that covers the maximum number of strings in* A *with up to* λ *mismatches.*

In real MP-PCR experiments, it may not be possible to cover sufficiently many sequences with one primer. In practice, we want to design a minimum set of primers with a threshold on the degeneracy, to cover all target sequences. This problem is called *Multiple Degenerate Primer Design*(**MDPD**). In **MDPD**, we can put the degeneracy threshold on either each single primer or the summation of whole set, called *Primer-Threshold* ***MDPD*** (**PT-MDPD**) and *Total-Threshold* ***MDPD***(**TT-MDPD**), respectively. These two problems are defined as in Problem 4 and Problem 5. These problems have been studied and proved to be NP-hard by Souvenir.

Problem 4 (PT-MDPD) *Given a set of n target strings* $A = \{a^1, a^2, \ldots, a^n\}$ *over alphabet* Σ *each of length* m*, and an integer* d*, find a minimum set of degenerate primers of length* m *and each primer is of degeneracy at most* d *that covers all strings in* A*.*

Problem 5 (TT-MDPD) *Given a set of n target strings* $A = \{a^1, a^2, \ldots, a^n\}$ *over alphabet* Σ *each of length* m*, and an integer* d*, find a minimum set of degenerate primers of length* m *and the total degeneracy is at most* d *that covers all strings in* A*.*

All degenerate primer design problems above can be reduced to regular primer design problems by setting the degeneracy threshold d to 0.

3.2 Previous Work

MDPD is closely related to the practical primer design process and has been thoroughly studied. Since it is NP-hard, it has no polynomial-time exact algorithm. However, several heuristic algorithms - such as HYDEN [41], MIPS [49], DPS [15] and DPS-HD [14] - were proposed.

HYDEN Proposed by Linhart and Shamir, this algorithm solves MDPD by repeatedly running beam search. Beam search is a heuristic search algorithm that explores a graph of partial solutions by expanding the most promising vertices. Here, the algorithm uses a beam to store a sufficient number of best primer candidates and then extends from this set of candidates.

MIPS and DPS follow the iterative beam search concept in HYDEN, but use different criteria to either choose or extend the "good" candidates so the performance of these algorithms depends on the beam size. DPS-HD tried to get rid of the dependency on the beam size by using Hamming distances and randomization.

MIPS Let a *c-primer* be a degenerate primer that covers c target sequences. Given n sequences of length l to design a set of primers of length m with maximum degeneracy d, MIPS runs the following process repeatedly until all sequences are covered: it starts with a set of 2-primers. In each iteration, each primer candidate extends its coverage by 1, introducing more degeneracy if necessary. In other words, in each iteration, MIPS generates $(c+1)$-primers from c-primers. A subset of primers is chosen as the new primer candidate set for next iteration.

MIPS scores these candidate primers by degeneracy and keeps b best candidates in a beam. It takes $O(bln)$ time to update the beam and check the coverage of b candidates against all $O(ln)$ substrings of length m of target sequences, which are also called m-mers. In a worst case scenario, the beam will be updated $O(n)$ times since there are n sequences to cover. So, the time complexity for updating the beam is $O(bln^2)$.

If the best primer set contains t primers, the beam-search process will repeat $O(tn)$ times. The total time complexity of MIPS is $O(bln^2 \times nt) = O(bln^3t)$.

DPS The DPS algorithm was proposed by Balla *et al.*. They adopt the beam search concept from MIPS, but use different scoring criteria and extension rules.

Much like MIPS, DPS starts with a set of 2-primers. In each iteration, b best primes are extended to a new candidate set by merging them with all possible m-mers in uncovered sequences. This new candidate set is used to start the next iteration. Once the degeneracy of the best candidate reaches the threshold d, we start a new round from 2-primers and repeat this until all sequences are covered. In each round, one best candidate is added into the final output set.

In each iteration, and much like in MIPS, DPS takes O(bln^2) time to sort and identify unique primers in the candidate set. The obvious difference between DPS and MIPS is that DPS's degeneracies are strictly and carefully increased in each iteration. If the threshold of degeneracy is d, the number of degenerate positions in primers is between $\log_{|\Sigma|} d$ and $\log_2 d$, and it takes $\log_2 d \sim (|\Sigma|-1)\log_{|\Sigma|} d$ iterations to reach this threshold. It takes $O(|\Sigma| \log_{|\Sigma|} d \times bln^2)$ to finish a round and pick one primer to the output set. If there are t primers in the output set, the total time complexity of DPS is $O(t \times |\Sigma| \log_{|\Sigma|} d \times bln^2) = O(b|\Sigma| \log_{|\Sigma|} dln^2t)$.

Compared to MIPS, DPS offers two important improvements. First, MIPS uses degeneracy as a scoring function. DPS, however, considers degeneracy and the coverage at the same time. Second, in DPS, the degeneracy of each primer candidate is strictly increased in each iteration, while, in MIPS, it may remain unchanged. This DPS property reduces the number of iterations from $O(n)$ to $O(|\Sigma| \log_\Sigma d)$.

DPS-HD and DPS-DIP Since MIPS and DPS use beam searches, their respective performances depend on the beam size b (i.e., the number of candidates in each iteration). Choosing b is very important for this kind of algorithm.

DPS-HD removes this kind of dependency on beam size by using randomization and Hamming distance. At first, it randomly chooses a sequence and uses all m-mers in this selected sequence as the candidate set. The Hamming distances between each candidate and all m-mers in other sequences are calculated. Each candidate merges with a randomly selected closest m-mer. Next, the distances from the new candidate to m-mers in uncovered

sequences are re-calculated until the degeneracy of the new candidate reaches threshold d or all sequences are covered. Then, the best candidate is added into the final output set. The size of the candidate set remains unchanged, so there is no need to set the beam size b to keep the best b candidates.

In one iteration, it takes $O(nl^2)$ time to calculate the Hamming distances between each candidate and all other m-mers. Just like in DPS, in DPS-HD, the degeneracy is strictly increased so the total number of iterations is $O(|\Sigma| \log_{|\Sigma|} d)$. The process can repeat $O(t)$ times where t is the number of primers in the output set so that the overall time complexity for DPS-HD is $O(|\Sigma| \log_{|\Sigma|} dnl^2t)$.

DPS-HD has a variant called DPS-DIP. The algorithms are basically the same, except that DPS-DIP uses *degenerate increase potential* (DIP) as the ranking criterion instead of the Hamming distance. The DIP between a primer candidate, p, and a m-mer, q, indicates how much the degeneracy will increase if p merges with q. For example, the DIP for the new degenerate primer candidate $p' = p \cup q$ is $f = \frac{deg(p')}{deg(p)}$.

DIP is based on the concept that when two m-mers have the same Hamming distance to a current candidate, we want to minimize the increasing of degeneracy. Take regular DNA sequences ($\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$), for example. When we add one extra symbol at a specific position, the Hamming distance is always changed by 1, while the DIP value may change by 2, $\frac{3}{2}$, or $\frac{4}{3}$, depending on how many symbols are originally at this position.

Let n be the number of target sequences, l be the length of each sequence, and suppose that we design t primers of length m with beam size b. Given the threshold d on the degeneracy, the following table compares both the time complexity and space complexity of the above three algorithms for **MDPD**:

	time complexity	space complexity
MIPS	$O(bln^3t)$	$O(n^2 + lmn)$
DPS	$O(\|\Sigma\| \log_{\|\Sigma\|} dbln^2t)$	$O(bln(n + \|\Sigma\|m))$
DPS-HD	$O(\|\Sigma\| \log_{\|\Sigma\|} dl^2nt)$	$O(l^2n)$

Experimental results show that DPS can cover input sequences with fewer primers than MIPS. About the running time: since DPS and MIPS algorithms are not implemented in the same programming language, the running time cannot be compared directly.

DPS-HD replaces the beam-search by randomization and users do not need to set the beam size. Although DPS can find a smaller set of primers, the result from DPS-HD

is very close, and DPS-HD takes much less time than DPS, especially when the number of input sequences is large.

3.3 Degenerate Primer Design and Bi-clique

As mentioned before, in this thesis, we focus on maximizing the coverage of a single degenerate primer (the **MCDPD** problem). To simplify the problem, we start from the binary case (i.e., $\Sigma = \{0, 1\}$; all target sequences and primers are binary strings). We can easily extend the binary case to more general case, such that $|\Sigma| = 4$, which is what we really need. In the binary case, we use the symbol '*' to indicate degenerate positions where both bit 0 and 1 are acceptable. **MCDPD** is an NP-hard problem, which can be proved by reducing the clique problem to it [41] . We notice that **MCDPD** is related highly to the bi-clique problem, as stated below. Thus, to analyze the approximability, we can interpret it in terms of bi-clique.

A bipartite graph $B = (U, V, E)$ is called a *bi-clique*, or a *complete bipartite graph* if $E = \{(u, v) : \forall u \in U, v \in V\}$. It can be denoted by $U \times V$ without explicitly specifying E. **MCDPD** can be formulated in terms of computing bi-cliques in bipartite graphs, as stated below. Consider an **MCDPD** instance that consists of a set A of n target sequences of length m and degeneracy threshold d. For a target sequence $a^i \in A$, let a^i_j be the j^{th} bit of a^i. We can construct a bipartite graph $G = (U, V, E)$ from A as follows:

- For each position $j = 1, \ldots, m$, we construct two corresponding vertices u^0_i and u^1_i in V_1, representing bit 0 or 1 at the i^{th} position of the primer. By this construction, $|U| = 2m$.
- For each target sequence $a^i \in A$, we construct a corresponding vertex v^i in V. Thus, $|V| = n$.
- If $a^i_j = 0$ then $(u^0_j, v^i) \in E$ but $(u^1_j, v^i) \notin E$, and vice versa if $a^i_j = 1$.

We give an example for $m = 3$ and $n = 4$:

$$S = \{101, 110, 001, 100\}$$

We construct vertex v^1 for sequence 101, v^2 for 110, v^3 for 001 and v^4 for 100. The

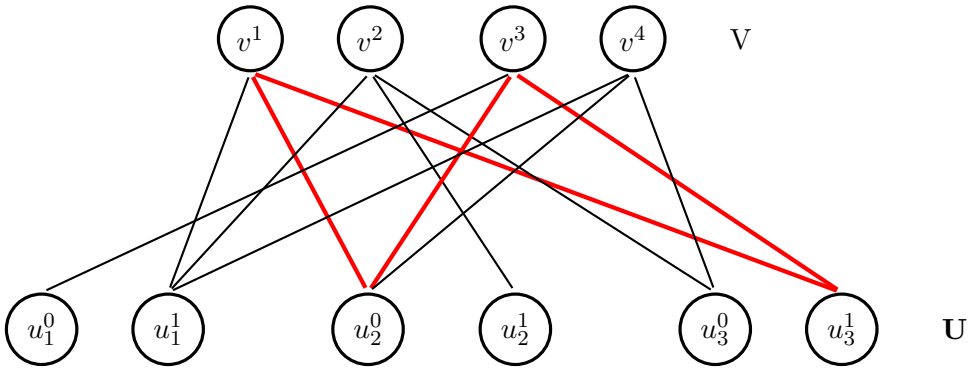


Figure 3.1: Bipartite graph G constructed from S.

corresponding bipartite graph is shown in Figure 3.1. The bold red lines show a bi-clique $\{u_2^0, u_3^1\} \times \{v^1, v^3\}$.

By this reduction, G contains a bi-clique $\widehat{U} \times \widehat{V}$ with $|\widehat{U}| = \delta$ and $|\widehat{V}| = c$ if, and only if, there exists a primer p with degeneracy $deg(p) \leq 2^{(m-\delta)}$ and p covers at least c target sequences. In other words, p has δ non-degenerate bits and can hybridize to at least c sequences. In Figure 3.1, the bi-clique $\{u_2^0, u_3^1\} \times \{v^1, v^3\}$ corresponds to a primer $p = *01$ of degeneracy $2^{3-2} = 2$.

This transformation shows that we can express **MCDPD** as a problem of computing a bi-clique $\widehat{U} \times \widehat{V}$ with $|\widehat{U}| \geq m - \log d$ and maximum $|\widehat{V}|$ in the constructed graph.

Notice that the reduced bi-clique problem has the following property below:

(1) The set U is partitioned into pairs $\{u_i^0, u_i^1\}$. For each pair, every vertex in V is adjacent to exactly one vertex.

(2) For any bi-clique $\widehat{U} \times \widehat{V}$, it picks at most one vertex from each pair $\{u_i^0, u_i^1\}$. This property is actually implied by property (1).

MCDPD is a maximization problem which has been proven to be NP-hard [41]. We are interested in designing approximation algorithms for it or proving lower bounds on the approximation ratio. Given an approximation algorithm *Alg* for a maximization problem Π, for an instance X, let $Opt(X)$ be the optimal solution of X and $Alg(X)$ be the solution obtained by *Alg*. We call *Alg* an α-approximation algorithm if for any instance X, *Alg* guarantees that $Opt(X)/Alg(X) \leq \alpha$ where $\alpha \geq 1$. In other words, algorithm *Alg* has

approximation ratio α. When α is smaller, the solutions computed by *Alg* are closer to the optimum.

Combining two different methods, a $2^{m/2}$-approximation algorithm for **MCDPD** has been proposed in [41]. When $\delta < \frac{m}{2}$, they designed the degenerate primer using the following algorithm: from the 1^{st} to the δ^{th} position, one by one, we pick the bit 0 or 1 that gives higher coverage. The remaining positions are degenerate. This way, the first bit of the primer matches at least $\frac{1}{2}$ of target sequences and the second bit matches at least half of these matched sequences, which is a fraction $\frac{1}{4}$ of the target sequences. Overall, we can obtain a primer that covers at least the fraction $\frac{1}{2^\delta}$ of target sequences. The optimal coverage is at most the number of target sequences, thus this algorithm has approximation ratio $2^\delta \leq 2^{\frac{m}{2}}$.

When $\delta \geq \frac{m}{2}$, we can simply examine all target strings and pick the one with maximum number of occurrences. Assume that the optimal solution q can cover c target sequences with degeneracy d. Since q is composed of at most d regular primers, at least one of them covers the fraction $\frac{c}{d}$ of the target sequences. We know that p has the highest coverage among all regular primers, so p also covers at least $\frac{c}{d}$ of the target sequences. This algorithm has approximation ratio $d = 2^{m-\delta} \leq 2^{\frac{m}{2}}$.

So far, we have not been able to design an approximation algorithm for **MCDPD** with a significantly better ratio than the one in [41], nor to prove any lower bound. To gain more insight, we have examined other versions of the bi-clique problem obtained by relaxing some constraints in **MCDPD**.

- The *threshold paired bi-clique* problem (**ThrPairBiClique**) drops the property (1) of bipartite graph G in **MCDPD**. In **ThrPairBiClique** problem, the G can have arbitrary edges.

- Besides removing property (1), the *paired bi-clique* problem (**PairBiClique**) further puts the constraint on the size of $|\widehat{U}|$. We can choose as much vertices as $\widehat{U}$ unless at most one vertex is chosen from each pair.

- In *threshold bi-clique* (**ThrBiClique**), both properties are dropped. G can have arbitrary edges and we simply find a bi-clique with $|\widehat{U}| \geq \delta$ and maximum $\widehat{V}$.

- The *maximum bi-clique* (**MaxBiClique**) problem drops both properties and also the constraint on $|\widehat{U}|$. We simply maximized the size of bi-clique(i.e., $|\widehat{U}| + |\widehat{V}|$).

These relationships are shown in Figure 3.2. In the following sections, we will give the formal definition and discuss the **MaxBiClique**, **ThrBiClique** and **ThrPairBiClique** problems. We have not yet investigated the **PairBiClique** problem and we do not know if it can be solved in polynomial time.

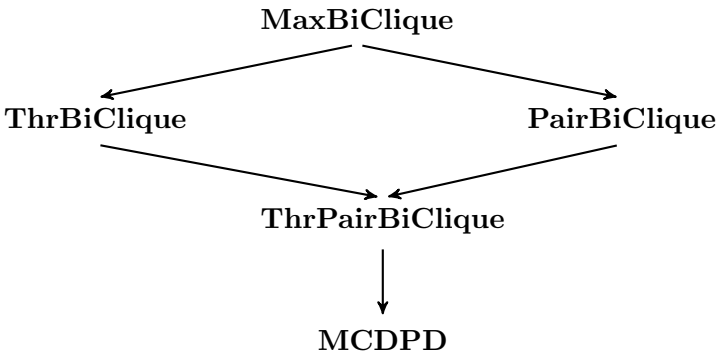


Figure 3.2: Relationships between bi-clique problems and **MCDPD**

3.3.1 MaxBiClique

In the **MaxBiClique** problem, we are given a bipartite graph $G = (U, V, E)$ and the objective is to find a complete bipartite subgraph $B = \widehat{U} \times \widehat{V}$ in G such that $\widehat{U} \subseteq U$, $\widehat{V} \subseteq V$, and $|\widehat{U}|+|\widehat{V}|$ is maximized. **MaxBiClique** has a trivial 2-approximation algorithm: we pick the vertex v with maximum degree z in G and all its adjacent vertices as a bi-clique of $1 + z$ vertices. Since v has the maximum degree, we know that $|\widehat{U}| \leq z$ and $|\widehat{V}| \leq z$ and, thus, $2(z + 1) \geq |\widehat{U}| + |\widehat{V}|$.

MaxBiClique is closely related to two other well-known optimization problems: Independent Set and Vertex Cover. Consider the complementary graph $\overline{G} = (U, V, \overline{E})$ of G. If $\widehat{U} \times \widehat{V}$ is a bi-clique in G, then $\widehat{U} \cup \widehat{V}$ will form an independent set in $\overline{G}$. This implies that $(U \cup V) - (\widehat{U} \cup \widehat{V})$ is a vertex cover of $\overline{G}$ because no edge exists within $\widehat{U} \cup \widehat{V}$ w.r.t. $\overline{G}$.

From the above, we can obtain a maximum bi-clique in G from a minimum vertex cover C in $\overline{G}$. Minimum Vertex cover and Maximum matching are dual problems. In a bipartite graph, they have the same cardinality and we can convert a maximum matching to a minimum vertex cover.

We can examine the relationship between these problems to design a polynomial time algorithm for **MaxBiClique**.

Algorithm 6 (MaximumBiclique$\big(G = (U, V, E)\big)$)

Construct the complementary graph $\overline{G} = (U, V, \overline{E})$

Find a maximum matching M *in* $\overline{G}$

Let L *be the set of nodes reachable from* $U - M$ *by alternating paths with respect to* M

Let $C = (U - L) \cup (V \cap L)$ ▷ *C is a minimum vertex cover*

return $(U \cup V) - C$

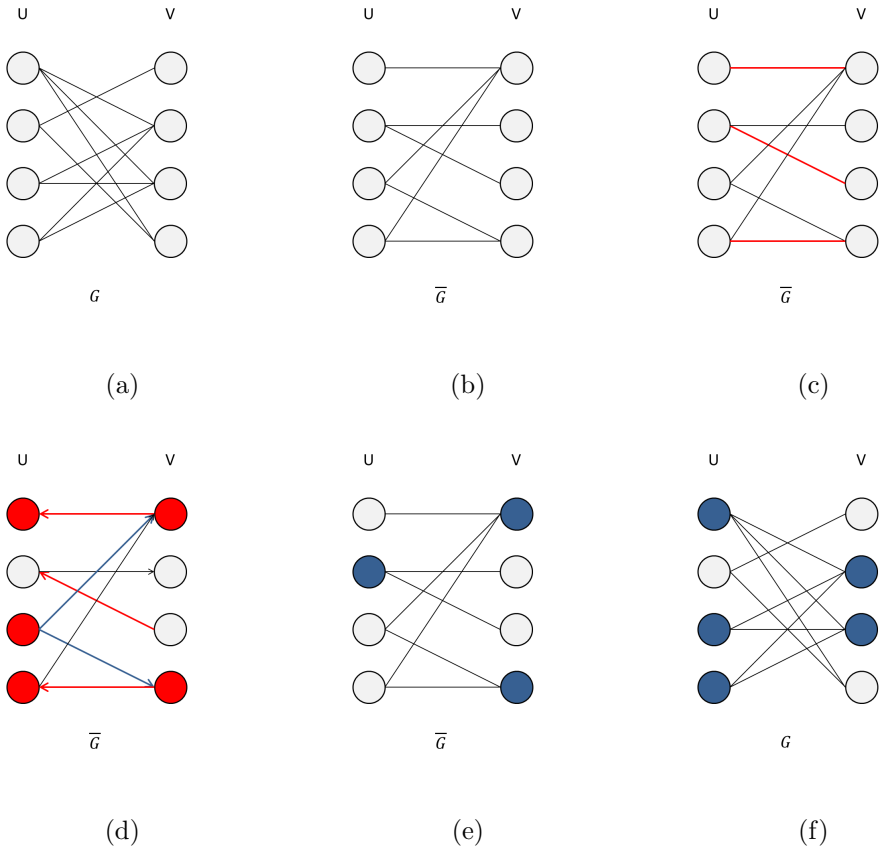


Figure 3.3: Execution of Algorithm 5: (3.3a) Input graph G. (3.3b) Complementary graph $\overline{G}$. (3.3c) A matching M in $\overline{G}$. (3.3d) Dark-shaded vertices are the set L w.r.t M. (3.3e) Minimum vertex cover C in $\overline{G}$: $(U - L) \cup (V \cap L)$. (3.3f) Maximum bi-clique B in G : $(U \cup V) - C$.

Figure 3.3 gives an example demonstrating how Algorithm 5 works. Given the bipartite graph G in Figure 3.3a, it shows the result of each step and the bi-clique obtained by this algorithm.

Assume the input graph G contains n vertices and m edges. A maximum matching M in $\overline{G}$ can be computed in time $O(\sqrt{n} \cdot |\overline{E}|) = O(n^{2.5})$ by the HopcroftKarp algorithm [28]. This algorithm will also compute the set L. In step 4, we convert the maximum matching M into a minimum vertex cover C. This step takes time $O(n)$, since we already have the set L from the previous step. The overall time complexity of Algorithm 5 is thus $O(n^{2.5})$.

Instead of using a reduction to the maximum matching problem, we can also formulate **MaxBiClique** as an integer linear program and solve it directly. Given a bipartite graph $G = (U, V, E)$, for each vertex $v \in U \cup V$, we introduce a variable x_v. If v is chosen in the maximum bi-clique then $x_v = 1$, else $x_v = 0$. The constraints are that if two vertices u, v are both chosen in the bi-clique, then the edge (u, v) must exist in G. The integral linear program ILP_1 for **MaxBiClique** is as follows:

$$
\begin{aligned}
\max \quad & \sum_{v \in U \cup V} x_v & & & & (ILP_1) \\
\text{subject to} \quad & x_u + x_v & \leq & \quad 1 & \forall u \in U, v \in V, \text{ and } (u, v) \notin E \\
& x_i & \in & \{0, 1\} & \forall i = 1, \ldots, n
\end{aligned}
$$

The constraint matrix A of ILP_1 is the incidence matrix of the complementary graph $\overline{G}$ in which rows correspond to edges in $\overline{G}$ and columns correspond to vertices.

For example, the incidence matrix A of the graph $\overline{G}$ in Figure 3.3a is

$$
A = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1
\end{bmatrix}
$$

$\overline{G}$ is also a bipartite graph, so its incidence matrix A is totally unimodular [58]. Total unimodularity allows us to drop integral constraints because ILP_1's relaxation - a linear

program - has an integral optimal solution, which is also a solution for the original integral linear program ILP_1. Thus, the **MaxBiClique** problem can be solved in polynomial time.

3.3.2 PairBiClique

Problem Definition A paired bi-clique B is a restricted bi-clique in a bipartite graph $G = (U, V, E)$ that has special structures. Vertices in U must be paired and B can contains at most one vertex from each pair. The *Max Paired Bi-clique problem* (**PairBiClique**) is defined as follows:

Problem 7 (MaxPairBiClique) *Given a bipartite graph $G = (U, V, E)$ in which vertices in U are partitioned into $\frac{|U|}{2}$ disjoint pairs $c_i = \{u_i, \overline{u_i}\}$, for $i = 1, \ldots, \frac{|U|}{2}$. Find a bi-clique $\widehat{U} \times \widehat{V}$ in G such that $|\widehat{U}| + |\widehat{V}|$ is maximum and $\widehat{U}$ contains at most one vertex from each pair.*

The decision version of **MaxPairBiClique** decides if there is a paired bi-clique $B = \widehat{U} \times \widehat{V}$ of specific size δ in the given graph G, the definition of which is as follows:

Problem 8 (PairBiClique) *Given a bipartite graph $G = (U, V, E)$ in which vertices in U are partitioned into $\frac{|U|}{2}$ disjoint pairs $c_i = \{u_i, \overline{u_i}\}$ for $i = 1, \ldots, \frac{|U|}{2}$ and an integer δ. Decide if there is G contains a bi-clique $\widehat{U} \times \widehat{V}$ in G such that $|\widehat{U}| + |\widehat{V}| = \delta$ and $\widehat{U}$ contains at most one vertex from each pair.*

NP-completeness The **PairBiClique** problem is NP-complete. It is easy to see that **PairBiClique** $\in$ NP, since a non-deterministic algorithm can guess a δ-element subset of vertices that contains at most one vertex from each pair in V_1, can check in polynomial time if that subset is a bi-clique or not. The NP-hardness can be proved by the reduction from the CLIQUE problem.

Theorem 9 ***PairBiClique*** *is NP-hard.*

Proof. Given an instance $\langle G, k\rangle$ of CLIQUE where $G = (V, E)$ and $|V| = n$, we can construct an instance of **PairBiClique** $\langle G', n + k\rangle$ from it such that G contains a clique of size k $\iff$ G' and contains a paired bi-clique of size $|\widehat{U}| + |\widehat{V}| \geq n + k$. The bipartite graph $G' = (U', V', E')$ is constructed as follows:

- We construct a pair of vertices $\{u_x, \overline{u}_x\}$ in U' and a vertex v_x in V' for each $x \in V$.

- The set of edges E' is defined as follows:

  1. If $(x, y) \in E$ or $x = y$, then $(u_x, v_y) \in E'$ and $(u_y, v_x) \in E'$.
  2. For any $x \neq y$, $(\overline{u}_x, v_y) \in E'$.

In this construction, we have $|U| = 2n$ and $|V| = n$. The vertices in U are paired. We now prove that the following statement is true:

G contains a clique of size $k \Longleftrightarrow G'$ contains a paired bi-clique of size $n + k$.

($\Rightarrow$) Suppose there is a clique C of size k in the graph G. We can choose the bi-clique $\widehat{U} \times \widehat{V}$ in G' as follows: $\widehat{U} = \{u_x | x \in C\} \cup \{\overline{u}_x | x \notin C\}$, and $\widehat{V} = \{v_y | y \in C\}$. This way we have $|\widehat{U}| = n$ and $|\widehat{V}| = k$.

For vertices $u_x \in \widehat{U}$ and $v_y \in \widehat{V}$, since $x, y \in C$, we know that it is either $x = y$ or $(x, y) \in E$. Thus, $(u_x, v_y) \in E'$. For $\overline{u}_x \in \widehat{U}$, since v_x is not chosen in $\widehat{V}$, it connects to every vertex $v_y \in \widehat{V}$. Ultimately, then, $\widehat{U} \times \widehat{V}$ is a bi-clique.

($\Leftarrow$) Let $B_1 = U_1 \times V_1$ be a bi-clique in G' of size $n + k$. We show that B_1 can be adjusted to a bi-clique$B_2 = U_2 \times V_2$ with $|U_2| = n$ and $|V_2| = k$. We can further obtain a **clique** C in the graph G from this bi-clique B_2.

Assume that the bi-clique B has $|U_1| < n$. Because U is composed of n pairs of vertices, there must exist some pair $\{u_x, \overline{u}_x\}$ such that $u_x, \overline{u}_x \notin U_1$. We can obtain a new bi-clique $B' = U_1' \times V_1'$ such that $|U_1'| = |U_1| + 1$ and $|V_1'| = |V_1| - 1$ by modifying B as below. For a pair $\{u_x, \overline{u}_x\} \notin U_1$, if v_x is in V_1, then let $V_1' = V_1 - v_x$, otherwise, we choose an arbitrary vertex like v_y and let $V_1' = V_1 - v_y$. Since $\overline{u_x}$ connects to every vertex $v_y \in V_1 - v_x$ and now v_x is not in V_1', we can add u_x into U_1, i.e., $U_1' = U_1 \cup u_x$, and $U_1' \times V_1'$ is still a bi-clique. We repeat this process until obtaining a bi-clique B_2 such that $|U_2| = n$ and $|V_2| = k$.

After obtaining B_2, we can further obtain a clique C in G of size k from it. Since $|U_2| = n$, we know that U_2 contains exactly one vertex from each pair $\{u_x, \overline{u}_x\}$. If $v_x \in V_2$, then it is always true that $\overline{u}_x \notin U_2$ and $u_x \in U_2$, since $(\overline{u}_x, v_x) \notin E'$. Or, if $v_x \notin V_2$ and $u_x \in U_2$, then we can remove u_x and add $\overline{u}_x$ to obtain a new bi-clique of the same size. Now, we have a bi-clique in which $u_x \in U_2$ if, and only if, $v_x \in V_2$. Let C be the set of all

x which $u_x \in U_2$ and $v_x \in V_2$, then C forms a clique of size k in G. To complete the proof, it is sufficient to show that the above statement is true. ■

It is trivial that **MaxPairBiClique** and **PairBiClique** are equivalent, thus, from above, we know both problems are NP-complete.

Approximability In this section, we prove that **MaxPairBiClique** $\in$ APX by providing a 1.5 approximation algorithm for **MaxThrBiClique**.

Given a instance of **MaxPairBiClique** , $G = (U, V, E)$, where U is paired. We know that U can be present as $U = X \cup \overline{X}$. For each $u_x \in X$, there exists a vertex $\overline{u}_x \in \overline{X}$, where $\{u_x, \overline{u}_x\}$ is a pair.
Also, $E = E' \cup \overline{E}'$ where E' is the set of edges between X and V, and $\overline{E}'$ is the set of edges between $\overline{X}$ and V .

The following is an approximation algorithm for the **PairBiClique** problem:

Algorithm 10 (PairedBiclique$\big(G = (U, V, E)\big)$)

1: $A_X \times B_X \leftarrow$MaximumBiclique$\big((X, V, E')\big)$.
2: $A_{\overline{X}} \times B_{\overline{X}} \leftarrow$MaximumBiclique$\big((\overline{X}, V, \overline{E}')\big)$.
3: ***return*** $\max\{A_X \times B_X, A_{\overline{X}} \times B_{\overline{X}}\}$

Now we proof the approximation ratio:

Theorem 11 *Algorithm 10 is a 1.5-approximation algorithm for* ***MaxPairBiClique*** *problem.*

Proof. In G, let $U^* \times V^*$ be the real optimal solution for **MaxPairBiClique**, where $|U^*| = a$ and $|V^*| = b$, U^* may contain vertices from both X and $\overline{X}$. Let $a' = |U^* \cap X|$ and $a'' = |U^* \cap \overline{X}|$ so we have $a = a' + a''$.

The maximum bi-clique in any bipartite graph can be computed in polynomial time as stated in section 3.3.1. In step 1 and 2, we will obtain two bi-cliques, $A_X \times B_X$ and $A_{\overline{X}} \times B_{\overline{X}}$, by calculating maximum bi-cliques that containing only X and $\overline{X}$, respectively. Let $|A_X| = a_1$, $|B_X| = b_1$, $|A_{\overline{X}}| = a_2$, and $|B_{\overline{X}}| = b_2$.

Since $A_X \times B_X$ is the maximum bi-clique using only X, we have $a' + b \leq a_1 + b_1$, also, $a'' + b \leq a_2 + b_2$. From above two inequalities, we know that

$$(a_1 + b_1) + (a_2 + b_2) \geq a' + b + a'' + b = a + 2b.$$

So, $\max(a_1 + b_1, a_2 + b_2) \geq \frac{1}{2}(a + 2b) = \frac{1}{2}a + b.$. Thus, the solution returned by above algorithm has size at least $\frac{1}{2}a + b$. Also, since we can always use X or $\overline{X}$ as a bi-clique, we have $a_1 + b_1 \geq \frac{|U|}{2}$ and $a_2 + b_2 \geq \frac{|U|}{2}$.

Assume that the solution from the above algorithm has size *AppSol*. Our goal is to find, then, the upper bound of $\frac{a+b}{AppSol}$. Now we consider the relationship between $\frac{1}{2}a$ and b. If $\frac{1}{2}a \leq b$, then we have

$$\frac{a+b}{AppSol} \leq \frac{a+b}{\frac{1}{2}a+b} \leq \frac{3}{2}.$$

We then have $\frac{1}{2}a < b$. Since $\frac{|V_1|}{2} \geq a$, we know that $AppSol \geq a$. Thus,

$$\frac{a+b}{AppSol} \leq \frac{a+b}{a} \leq \frac{3}{2}.$$

1.5 is an upper bound for $\frac{a+b}{AppSol}$. Therefore, the above approximation algorithm has approximation ratio 1.5. ■

Currently, we know that the **MaxPairBiClique** problem doesn't have FPTAS since the size of paired bi-clique is an integer bounded by the size of input graph. However, we do not yet have a lower bound for the approximation ratio.

3.3.3 ThrBiClique

Definition The *Max Threshold Bi-clique problem* (**MaxThrBiClique**) is defined as follows:

Problem 12 (MaxThrBiClique) *Given a bipartite graph $G = (U, V, E)$ and a positive integer $\delta \leq |U|$. Find a bi-clique $\widehat{U} \times \widehat{V}$ such that $|\widehat{U}| = \delta$ and $|\widehat{V}|$ is maximum.*

The decision version of **MaxThrBiClique** is to find if there is a bi-clique of specific size $\delta \times l$ in a given graph G:

Problem 13 (ThrBiClique) *Given a bipartite graph $G = (U, V, E)$ and two positive integers $\delta \leq |U|$ and $l \leq |V|$. Decide if G contains a bi-clique $\widehat{U} \times \widehat{V}$ such that $|\widehat{U}| = \delta$ and $|\widehat{V}| = l$.*

NP-completeness The **MaxThrBiClique** problem can be formulated as the *Maximum Fixed Set Intersection* problem, which is known as a NP-complete problem [21].

Given an instance of **ThrBiClique**, we can construct a set S_u for each vertex $u \in U$ and an element e_v for each vertex $v \in V$. If $(u, v) \in E$, then $e_v \in S_u$. Therefore, computing a bi-clique $\widehat{U} \times \widehat{V}$ with $|\widehat{U}| = \delta$ and maximum $|\widehat{V}|$ is equivalent to finding δ sets whose intersection is maximized. From above, it is clear that **MaxThrBiClique** and **ThrBiClique** are NP-complete problems.

Threshold bi-clique and Balanced bi-clique We note that threshold bi-clique problems are highly related to balanced bi-clique problems, which are well-studied. The **BalancedBiclique** problem and its optimization version are defined as follows, in which it is a known NP-hard problem [27].

Problem 14 (BalancedBiclique) *Given a bipartite graph $G = (U, V, E)$ and a positive integer $\delta \leq |U|$, decide if G contains a bi-clique $\widehat{U} \times \widehat{V}$ in G such that $|\widehat{U}| = |\widehat{V}| = \delta$.*

Problem 15 (MaxBalancedBiclique) *Given a bipartite graph $G = (U, V, E)$, find a bi-clique $\widehat{U} \times \widehat{V}$ in G such that $|\widehat{U}| = |\widehat{V}|$ and $|\widehat{U}| + |\widehat{V}|$ is maximum.*

The **ThrBiClique** problem and the **BalancedBiclique** problem are equivalent in that if we can solve one, we can solve the other. It is obvious, then, that solving **ThrBiClique** implies solving **BalancedBiclique**. Likewise, we can always reduce a **ThrBiClique** instance to a **BalancedBiclique** instance by adding extra vertices.

The optimization versions, **MaxBalancedBiclique** and **MaxThrBiClique**, also have the same approximation ratio (i.e., if we have an algorithm to solve one problem within ratio α, then we can solve another with ratio α, too).

Theorem 16 *For any $\alpha > 1$, **MaxThrBiClique** has an α approximation algorithm $\Longleftrightarrow$ **MaxBalancedBiclique** has an α approximation algorithm.*

Proof. ($\Rightarrow$) Suppose we have an α approximation algorithm, $\mathcal{A}(G, \delta)$, for **MaxThrBiClique**. Given a graph $G = (U, V, E)$, we can find a balanced bi-clique by running $\mathcal{A}$ iteratively as in Algorithm 17.

Algorithm 17 (Approximate MaxBalancedBiclique$\big(G = (U, V, E)\big)$)

$\delta \leftarrow 0$, $l \leftarrow 0$, $B = \emptyset$, $B^* \leftarrow B$.

while $\delta \leq l$ ***do***

$B^* \leftarrow B$

$\delta \leftarrow \delta + 1$

$B = \widehat{U} \times \widehat{V} \leftarrow \mathcal{A}(G, \delta)$.

$l \leftarrow |\widehat{V}|$

return B^*

Assume that the optimal balanced bi-clique in G is of size t on both sides. Since there is a $t \times t$ bi-clique in G, for any parameter $\delta \leq t$, the optimal solution for **MaxThrBiClique** will be at least t. Thus, when $\delta = \frac{t}{\alpha}$, we will have $\mathcal{A}(G, \delta) \geq t \times \frac{1}{\alpha} = \frac{t}{\alpha} = \delta$. According to Algorithm 17, it will return a value at least $\frac{t}{\alpha}$, so it is an α-approximation algorithm for **MaxBalancedBiclique**.

($\Leftarrow$) W.l.o.g, we assume that the optimal bi-clique for **MaxThrBiClique** is of size $\delta \times l$ where $l \geq \delta$.

Suppose that we have an α approximation algorithm, $\mathcal{B}(G)$, for **MaxBalancedBiclique**. Given a graph $G = (U, V, E)$ and a positive integer δ, we can find a bi-clique that contains exactly δ vertices in U by running $\mathcal{B}$ iteratively as in Algorithm 18.

Algorithm 18 (Approximate MaxThrBiClique$\big(G = (U, V, E), \delta\big)$)

$k = \delta$, $W \leftarrow \emptyset$, $B \leftarrow \emptyset$.

while *1* ***do***

$B = \widehat{U} \times \widehat{V} \leftarrow \mathcal{B}(G)$

if $|\widehat{V}| < \frac{k}{\alpha}$ ***then***

return $B - W$

create a new vertex u

$U \leftarrow U \cup u$, $W \leftarrow W \cup u$

for $v \in V$ ***do***

$E \leftarrow E \cup (u, v)$

$k++$

Without loss of generality, assume that the optimal threshold bi-clique is of size $\delta \times l$ such that $\delta \leq l$. When Algorithm 18 terminates, let the optimal balanced bi-clique

have size $t \times t$ so we can obtain a balanced bi-clique of size $|\widehat{U}| = |\widehat{V}| \geq \frac{t}{\alpha}$. At this time, we have $\frac{k}{\alpha} > |\widehat{V}|$, so, it is clear that $k > t$. This result implies $l \leq t$, otherwise it contradicts that the maximum balanced bi-clique has size $t \times t$, so we have $|\widehat{V}| \geq \frac{l}{\alpha}$.

After removing the newly-added vertices from $\widehat{U}$, we can obtain a bi-clique, $(\widehat{U} - W) \times \widehat{V}$, where $|\widehat{U} - W| = \delta$ and $|\widehat{V}| \geq \frac{l}{\alpha}$. ■

From above, the complexity result for **MaxBalancedBiclique** is extremely helpful for **MaxThrBiClique**. The inapproximability of **MaxBalancedBiclique** is well-studied and the following theorem has been proven [35] :

Theorem 19 *Let $\epsilon > 0$ be an arbitrarily small constant. If SAT does not have a probabilistic algorithm that runs in time 2^{n^ϵ} on an instance of size n, then there is no polynomial time algorithm for **MaxBalancedBiclique** with approximation ratio $N^{\epsilon'}$ on graphs of size N where $\epsilon' = \frac{1}{2^{O(1/\epsilon \log(1/\epsilon))}}$.*

Therefore, we know that, for any small ϵ, if SAT does not have a probabilistic algorithm that runs in time 2^{n^ϵ}, there is no $N^{\left(\frac{1}{2^{O(1/\epsilon \log(1/\epsilon))}}\right)}$ approximation algorithm for **MaxThrBiClique**.

Linear Program and Integrality Gaps Intuitively, **MaxThrBiClique** can be formulated as an integer linear program. Given a bipartite graph (U, V, E), we introduce a variable x_u for each vertex $u \in U$, and y_v for $v \in V$. The linear program is as follows:

$$
\begin{array}{lrcll}
\max & \displaystyle\sum_{v \in V_2} y_v & & & (ILP_2) \\
\text{subject to} & \displaystyle\sum_{u \in V_1} x_u & = & \delta & \\
& x_u + y_v & \leq & 1 \quad \forall u \in U, v \in V, \text{ and } (u,v) \notin E & \\
& x_u, y_v & \in & \{0,1\} \qquad\qquad \forall i = 1, \ldots, n &
\end{array}
$$

Let $|U| = m$ and $|V| = n$. Without loss of generality, we can assume that $m > \delta$. In the relaxation of ILP_2, we can assign $x_u = \frac{\delta}{m}$ for all $u \in U$ and $y_v = 1 - \frac{\delta}{m}$ for all $v \in V$, so the fractional solution is $(1 - \frac{\delta}{m}) \cdot n$. Consider an empty graph G (e.g., $E = \emptyset$). The optimal integral solution will be $|\widehat{U}| = 0$, so the integrality gap is infinite.

Since $|\widehat{U}|$ is requi red to be at least δ, every vertex $v \in \widehat{V}$ has degree $deg(v) \geq \delta$. Thus, vertices in V with degree smaller than δ will never be chosen for $\widehat{V}$. So, we are only interested in graphs in which all vertices in V have degrees at least δ. With this constraint, we can construct a bipartite graph G as follows: let $m = \delta \cdot n$, and G consists of n disjoint stars of degree δ with centers in V. In this construction, the optimal solution in G is $|\widehat{V}| = 1$. The integrality gap is $(1 - \frac{\delta}{n \cdot \delta}) \cdot n = n - 1$. Therefore, for the bipartite graph with $deg(v) \geq \delta$, for every $v \in V$, the integrality gap is at least $n - 1$.

3.3.4 PairThrBiClique

Problem Definition and approximability Threshold and Paired **Bi-clique** problems have constraints on both the number of vertices and the paired property. The **MaxThrPairBiClique** problem is defined as follows:

Problem 20 (MaxThrPairBiClique) *Given a bipartite graph $G = (U, V, E)$ where vertices in U are partitioned into $\frac{|U|}{2}$ disjoint pairs $c_i = \{u_i, \overline{u_i}\}$ for $i = 1, \cdots, \frac{|U|}{2}$, and a positive integers $\delta \leq \frac{|U|}{2}$, decide if G contains a paired bi-clique $\widehat{U} \times \widehat{V}$ such that $|\widehat{U}| = \delta$ and at most one vertex is chosen from each pair.*

The **MaxThrPairBiClique** problem can be easily reduced to the **MaxThrBiClique** problem. This reduction not only proves the NP-completeness, but also provides some clue of the approximability of **MaxThrPairBiClique**. The reduction and the summary are as follows:

Theorem 21 ***MaxThrPairBiClique*** *is NP-complete and for any $\alpha > 1$, if* ***ThrPairBiClique*** *has an α-approximation algorithm, then* ***ThrBiClique*** *has an α-approximation algorithm.*

Proof. Given an instance (G, δ) of **ThrBiClique**, we can reduce it to an instance (G', δ) of **ThrPairBiClique** as follows: let $G = (U, V, E)$. We construct G' as $(U \cup \overline{U}, V, E)$ in which $\overline{U}$ is a set of independent vertices of size $|U|$. For each $u \in U$, there is a corresponding vertex $\overline{u} \in \overline{U}$. This way, one vertex, at most, in each pair $\{u, \overline{u}\}$ can be chosen for $\widehat{U}$.

This reduction makes clear the following relationship: G contains a bi-clique $\widehat{U} \times \widehat{V}$ with $|\widehat{U}| = \delta$ and $|\widehat{V}| = k$ if and only if G' contains a bi-clique $\widehat{U'} \times \widehat{V'}$ with $|\widehat{U'}| = \delta$ and $|\widehat{V'}| = k$, and in $\widehat{U'}$ one, at most, is chosen from each pair $\{u, \overline{u}\}$.

($\Rightarrow$) Consider that there is a bi-clique $\widehat{U} \times \widehat{V}$ in G, G' that contains the same bi-clique. $\overline{u}$ is not chosen for all $\overline{u} \in \overline{U}$.

($\Leftarrow$) Assume there is a bi-clique $\widehat{U'} \times \widehat{V'}$ in G'. $\widehat{U'}$ will not contain any vertex $\overline{u} \in \overline{U}$, since they are independent vertices and will cause $\widehat{V'} = \emptyset$. Therefore, $\widehat{U'} \subseteq U$ and $\widehat{U'} \times \widehat{V'}$ is also a bi-clique in G.

If there is an α-approximation algorithm A for **ThrPairBiClique**, then we can construct an α-approximation algorithm for **ThrBiClique** by running A on the graph G'. Thus, the inapproximability of **ThrBiClique** is an lower bound of the approximation ratio of **ThrPairBiClique**. ■

From the previous section, we know that for any small ϵ, if SAT does not have a probabilistic algorithm that runs in time 2^{n^ϵ}, there is no $N^{\left(\frac{1}{2^{O(1/\epsilon \log(1/\epsilon))}}\right)}$ approximation algorithm for **MaxThrBiClique**, and**MaxThrPairBiClique** also does not have any $N^{\left(\frac{1}{2^{O(1/\epsilon \log(1/\epsilon))}}\right)}$ approximation algorithm.

Chapter 4

Randomized Rounding Algorithm for Solving MCDPD$^{mis}_{tmpl}$

Problems **MCDPD** and **MCDPD**^{mis} are NP-complete (see Chapter 1.3). Therefore, polynomial time algorithms are not available unless $P = NP$. Here, we propose a heuristic algorithm to solve **MCDPD**^{mis}, utilizing randomized rounding approach. Recall that in the **MCDPD**^{mis}problem, we are given a collection A of strings over some alphabet Σ, in which each is the same length m, and an integral degeneracy threshold d as well as an integral mismatch allowance λ. The objective is to compute a degenerate primer p of length m and degeneracy at most d that covers the maximum number of strings in A with at most λ mismatches.

MCDPD is a special case of **MCDPD**^{mis} in which mismatches are not allowed (i.e., $\lambda = 0$). In this special case, an optimal primer p is guaranteed to cover at least one target string $a^i \in A$. Hence, p can be obtained by changing a target sequence a^i, adding possible bases at some positions to make them ambiguous bases without exceeding the degeneracy limit d.

From above, the **MCDPD** problem can be solved using the reduced problem 22, **MCDPD** with template(**MCDPD**$_{tmpl}$), trying each input sequence a^i as the template $\hat{p}$ and obtain the optimal result.

Problem 22 ($\mathbf{MCDPD}_{tmpl}$) *Given a set of n target strings $A = \{a^1, a^2, \ldots, a^n\}$ over alphabet Σ each of length m, an integer d, and a template string $\hat{p}$, find a degenerate primer p of length m with $deg(p)$ at most d that covers $\hat{p}$ and covers the maximum number of strings in A.*

Similarly, with mismatch allowance, $\mathbf{MCDPD}^{mis}$ can also reduced to the problem 23, $\mathbf{MCDPD}^{mis}$ with template($\mathbf{MCDPD}^{mis}_{tmpl}$). When mismatches are allowed to guarantee the optimal solution, not only the target strings a^i need to be tested as template. Let $Tmpl_\lambda(A)$ denote the set of strings over Σ of length m, which has Hamming distance smaller than λ to any target sequence. All strings in $Tmpl_\lambda$ are potential templates to obtain the optimal p.

We remark here that our algorithm for $\mathbf{MCDPD}^{mis}$ will not actually try all possible templates from $Tmpl_\lambda(A)$ - there are simply too many of these if λ is large. Instead, we randomly sample templates from $Tmpl_\lambda(A)$ and apply the algorithm for $\mathbf{MCDPD}^{mis}_{tmpl}$ only to those sampled. The number of samples will affect the running time and accuracy. See Chapter 4.4 for more details.

Problem 23 ($\mathbf{MCDPD}^{mis}_{tmpl}$) *Given A set of n target strings $A = \{a^1, a^2, \ldots, a^n\}$ over alphabet Σ each of length m, integers d and λ, and a template string $\hat{p}$, find a degenerate primer p of length m with $deg(p)$ at most d that covers $\hat{p}$ and covers the maximum number of strings in A with up to λ mismatches.*

We present our algorithm for $\mathbf{MCDPD}^{mis}_{tmpl}$ in 3 steps. In Section 4.1, we explain the fundamental idea of our approach by presenting the linear program and our randomized rounding algorithm for the case of binary strings, where $\Sigma = \{0, 1\}$. The extension to DNA strings is somewhat complicated due to the presence of several ambiguous bases. We present the algorithm for DNA strings in Section 4.2. At last the complete design process is described in Section 4.3

4.1 for Binary Case

In this section, we use binary alphabet to show the integer linear program representation of the $\mathbf{MCDPD}^{mis}_{tmpl}$ problem and demonstrate the randomized rounding algorithm. The binary alphabet set is $\Sigma = \{0, 1\}$ and we only have one ambiguous base, denoted by

N, which can represent either 0 or 1. The purpose of the randomized rounding algorithm, called SRR_{bin}, is to compute an optimal fractional solution of the linear program then gradually round it to a feasible integral solution.

Let $\hat{p} = \hat{p}_1\hat{p}_2\cdots\hat{p}_k$ be the template string from the given instance of $\mathbf{MCDPD}^{mis}_{tmpl}$. It is convenient to think of the objective of $\mathbf{MCDPD}^{mis}_{tmpl}$ as converting $\hat{p}$ into a degenerate primer p by changing up to δ symbols in $\hat{p}$ to N, where $\delta = \log_2 d$.

For each target string $a^i = a^i_1 a^i_2 \cdots a^i_k$, we use a binary variable x^i to indicate if a^i is covered by p. For each position j, a binary variable n_j is used to indicate if $\hat{p}_j$ will be changed to N. To take mismatch allowance into consideration, we also use binary variables μ^i_j, which indicate if we allow a mismatch between p and a^i on position j, that is, whether or not $a^i_j \not\subseteq p_j$.

By using the above variables, the objective is to maximize the sum of all x^i. Next, we need to specify the constraints. One constraint involves the mismatch allowance λ. For a string a^i, the number of mismatches $\sum_j \mu^i_j$ should not exceed λ. Next, we have the bound on the degeneracy. In the binary case, the degeneracy of p can be written as $deg(p) = \prod_j 2^{n_j}$ and we require that $deg(p) \leq d$. To convert this inequality into a linear constraint, we take the logarithms of both sides. The last group of constraints are the covering constraints. For each j, if p covers a^i and $\hat{p}_j \neq a^i_j$, then either $p_j = \text{N}$ or p_j contributes to the number of mismatches. These constraints can be expressed by inequalities $x^i \leq n_j + \mu^i_j$, for all i, j such that $a^i_j \neq \hat{p}_j$. The complete linear program is as such:

$$
\begin{array}{lrcll}
\max & \displaystyle\sum_i x^i & & & (ILP_3) \\
\text{subject to} & \displaystyle\sum_j \mu^i_j & \leq & \lambda & \forall i \\
& \displaystyle\sum_j n_j & \leq & \delta & \\
& x^i - n_j & \leq & \mu^i_j, \quad \forall i, j : \hat{p}_j \neq \text{N} \wedge a^i_j \neq \hat{p}_j & \\
& x^i, n_j, \mu^i_j & \in & \{0,1\} & \forall i, j
\end{array}
$$

The pseudo-code of our Algorithm SRR_{bin} is given below in Algorithm 24. The algorithm starts with $p = \hat{p}$ and gradually changes some symbols in p to N, solving a linear program at each step. At each iteration, the size of the linear program can be reduced by

discarding strings that are too different from the current p as well as by ignoring strings that are already matched by p. More precisely, any a^i which differs from the current p on more than $\lambda + \delta$ positions cannot be covered by any degenerate primer obtained from p, so this a^i can be discarded.

It it important to note that, if a^i differs from p with at most λ positions, then it will be covered, in which case we can both set $x^i = 1$ and remove it from A. This pruning process in Algorithm SRR_{bin} is implemented by function FilterOut.

Algorithm 24 (SRR_{bin})

$p \leftarrow \hat{p}$
while $deg(p) < d$ ***do***
  FilterOut(p, A, d, λ) ▷ *updates* A
  if $A = \emptyset$ ***then***
    break
  $LP \leftarrow$ GenLinProgram(p, A, d, λ)
  $FracSol \leftarrow$ SolveLinProgram(LP)
  RandRounding$_{bin}(p, FracSol, d)$ ▷ *updates* p
return p

If no sequences are left in A, then we are done and we can output p. Otherwise, we construct the linear program for the remaining strings. This linear program is essentially the same as Program ILP_3, with $\hat{p}$ replaced by p,and with new constraints $n_j = 1$ for positions j such that $p_j \neq \texttt{N}$.

Additional constraints are added to take into account the rounded positions in p. Namely, we add the constraint $n_j = 1$ for all p_j already replaced by N.

We then consider the relaxation of the above integer program in which all integral constraints $x^i, n_j, \mu_j^i \in \{0, 1\}$ are replaced by $x^i, n_j, \mu_j^i \in [0, 1]$, that is, all variables are allowed to take fractional values. After solving this relaxation, we call Procedure RandRounding$_{bin}$, which chooses one fractional variable n_j with probability proportional to its value and that rounds it up to 1. It is sufficient to round only the n_j variables, since all other variables are uniquely determined from the n_j's. To complete this task, let J be the set of all j for which $n_j \neq 1$ and $\pi = \sum_{j \in J} n_j$. The interval $[0, \pi]$ can be split into consecutive $|J|$ intervals, with the interval corresponding to $j \in J$ having length n_j. Thus,

we can randomly - but, uniformly - choose a value c from $[0, \pi]$. And, if c is in the interval corresponding to $j \in J$, then we round n_j to 1.

If the degeneracy of p is still below the threshold, Algorithm $\mathrm{SRR}_{\mathrm{bin}}$ executes the next iteration. It correspondingly adjusts the constraints of the linear program, which produces a new linear program. The process stops either when the degeneracy allowance is exhausted or when no target string can be further covered.

4.2 for DNA Sequences

The randomized rounding algorithm, $\mathrm{SRR}_{\mathrm{bin}}$, can be extended to apply on DNA sequences (i.e., when the alphabet is $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$).

We start with the description of the integer linear program for $\mathbf{MCDPD}^{mis}_{tmpl}$ with $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. Degenerate primers for DNA sequences - in addition to four nucleotide symbols `A`, `C`, `G` and `T` - can use eleven symbols corresponding to ambiguous positions described by the IUPAC codes `M`, `R`, `W`, `S`, `Y`, `K`, `V`, `H`, `D`, `B`, and `N`. The interpretation of these codes was given in Table 1.1 in Chapter 1.1. Let Ω denote the set of these fifteen symbols. Each symbol $\omega \in \Omega$ corresponds to a subset of Σ and $|\omega|$ is the for the cardinality of this subset. For example, we have $|\mathtt{C}| = 1$, $|\mathtt{H}| = 3$, and $|\mathtt{N}| = 4$. The complete linear program is given in program ILP_4.

As for binary sequences, x^i indicates if the i-th target sequence a^i is covered. The objective of the linear program is to maximize the primer coverage, that is $\sum_i x^i$. To specify the constraints, we now have eleven variables representing the presence of ambiguous bases in the degenerate primer, namely m_j, r_j, w_j, s_j, y_j, k_j, v_j, h_j, d_j, b_j, and n_j, which are denoted using letters corresponding to the ambiguous symbols. Specifically, for each position j and for each symbol $\omega \in \Omega$, the corresponding variable ω_j indicates if $\hat{p}_j$ is changed to this symbol in the computed degenerate primer p. For example, r_j represents the absence or presence of `R` in position j. For each j, at most one of these variables can be 1, which can be represented by the constraint that their sum is at most 1.

Variables μ_j^i indicate if there is a mismatch between p and a^i on position j. The bound on the number of mismatches can be written as $\sum_j \mu_j^i \leq \lambda$ for each i. The bound on the degeneracy of the primer p can be written as such

$$\deg(p) = \prod_j 2^{(m_j+r_j+w_j+s_j+y_j+k_j)} \times 3^{(v_j+h_j+d_j+b_j)} \times 4^{n_j} \ \leq \ d,$$

which, after taking logarithms of both sides, gives us another linear constraint.

$$\begin{array}{rll}
\text{maximize} & \sum_i x^i & (ILP_4) \\
\text{subject to} & \sum_j \mu_j^i \leq \lambda & \forall i \\
& \sum_j \big[(m_j + r_j + w_j + s_j + y_j + k_j) & \\
& \quad + \log_2 3 \cdot (v_j + h_j + d_j + b_j) & \\
& \quad + 2 \cdot n_j] \leq \delta = \log_2 d & \\
& x^i \leq m_j + v_j + h_j + n_j + \mu_j^i & \forall i,j : (\hat{p}_j = \texttt{A}, a_{ij} = \texttt{C}) \vee (\hat{p}_j = \texttt{C}, a_{ij} = \texttt{A}) \\
& x^i \leq r_j + v_j + d_j + n_j + \mu_j^i & \forall i,j : (\hat{p}_j = \texttt{A}, a_{ij} = \texttt{G}) \vee (\hat{p}_j = \texttt{G}, a_{ij} = \texttt{A}) \\
& x^i \leq w_j + h_j + d_j + n_j + \mu_j^i & \forall i,j : (\hat{p}_j = \texttt{A}, a_{ij} = \texttt{T}) \vee (\hat{p}_j = \texttt{T}, a_{ij} = \texttt{A}) \\
& x^i \leq s_j + v_j + b_j + n_j + \mu_j^i & \forall i,j : (\hat{p}_j = \texttt{C}, a_{ij} = \texttt{G}) \vee (\hat{p}_j = \texttt{G}, a_{ij} = \texttt{C}) \\
& x^i \leq y_j + h_j + b_j + n_j + \mu_j^i & \forall i,j : (\hat{p}_j = \texttt{C}, a_{ij} = \texttt{T}) \vee (\hat{p}_j = \texttt{T}, a_{ij} = \texttt{C}) \\
& x^i \leq k_j + d_j + b_j + n_j + \mu_j^i & \forall i,j : (\hat{p}_j = \texttt{G}, a_{ij} = \texttt{T}) \vee (\hat{p}_j = \texttt{T}, a_{ij} = \texttt{G}) \\
& x^i, m_j, r_j, w_j, s_j, y_j, k_j, & \\
& \quad v_j, h_j, d_j, b_j, n_j, \mu_j^i \in \{0,1\} & \forall i,j \\
& m_j + r_j + w_j + s_j + y_j + k_j & \\
& \quad + v_j + h_j + d_j + b_j + n_j \leq 1 & \forall j
\end{array}$$

For a^i to be covered (that is, when $x^i = 1$) in each position j for which $a_j^i \neq \hat{p}_j$, we must either have a mismatch at position j or we need $a_j^i \subseteq p_j$. Expressing this with linear constraints can be accomplished by considering cases corresponding to different values of $\hat{p}_j$ and a_j^i. For example, when $\hat{p}_j = \texttt{A}$ and $a_j^i = \texttt{C}$ (or, vice versa), then either we have a mismatch at position j (that is, $\mu_j^i = 1$) or p_j must be one of ambiguous symbols that match `A` and `C` (that is, `M`, `V`, `H`, or `N`). This scenario can be expressed by the constraint

$x^i \leq m_j + v_j + h_j + n_j + \mu_j^i$. We will have one such case for any two different choices of $\hat{p}_j$ and a_j^i, giving us six groups of such constraints.

We then extend our randomized rounding approach from the previous section to this new linear program. In this linear program, we can see that the integral solution can be determined based on the values of all variables ω_j, for $\omega \in \Omega$. In the fractional solution, a higher value of ω_j indicates that p_j is more likely to be the ambiguous symbol ω. We determine, therefore, ambiguous bases in p one at a time by rounding the corresponding variables.

As for binary strings Algorithm SRR_{dna}, will start with $p = \hat{p}$ and gradually change some bases in p to ambiguous bases, solving a linear program at each step. At each iteration we first call function FilterOut that filters out target sequences that are either too different from the template $\hat{p}$ (so that they cannot be matched), or too similar (in which case they are guaranteed to be matched).

Algorithm 25 (SRR_{dna})

$p \leftarrow \hat{p}$
while $deg(p) < d$ ***do***
  FilterOut(p, A, d, λ) ▷ *updates* A
  if $A = \emptyset$ ***then***
    break
  $LP \leftarrow$ GenLinProgram(p, A, d, λ)
  $FracSol \leftarrow$ SolveLinProgram(LP)
  RandRounding$_{dna}(p, FracSol, d)$ ▷ *updates* p *and* d
return p

If no sequences are left in A, then we output p and halt. Otherwise, we construct a linear program for the remaining sequences. This linear program is a slight modification of the one in Program ILP_4, with $\hat{p}$ replaced by p. Each base p_j that was rounded to an ambiguous symbol is essentially removed from consideration and will not be changed in the future. Specifically, the constraints on x^i associated with this position j will be dropped from the linear program because these constraints apply only to positions where $p_j \in \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$. For each position j that was already rounded, we appropriately modify the corresponding variables. If $p_j = \omega$ for some $\omega \in \Omega - \Sigma$, then the corresponding variable ω_j

is set to 1 and all other variables ω'_j are set to 0. If $a^i_j \in p_j$, that is, a^i_j is already matched, then we set $\mu^i_j = 0$. And, if $a^i_j \notin p_j$, then we set $\mu^i_j = 1$, which effectively reduces the mismatch allowance for a^i in the remaining linear program.

Next, Algorithm SRR_{dna} solves the fractional relaxation of such constructed integer program, obtaining a fractional solution *FracSol*.

Finally, the algorithm calls function $\text{RANDROUNDING}_{dna}$ that will round one fractional variable ω_j to 1. This represents setting p_j to ω. To choose j and the symbol ω for p_j, we randomly choose a fractional variable ω_j proportionally to its values among undetermined positions. This choosing is done similarly as in the binary case, by summing up fractional values that correspond to different symbols and positions and then choosing - uniformly - a random number c between 0 and this sum. This c determines which variable should be rounded up to 1.

4.3 Complete Algorithm

To assess the effectiveness of our randomized rounding approach, we extended Algorithm SRR_{dna} to a complete primer design algorithm - called RRD2P - and tested it experimentally on real data sets. In this section, we describe Algorithm RRD2P. The experimental evaluation is given in Chapter 4.4.

A flowchart of Algorithm RRD2P is given in Figure 4.1. The algorithm (see Algorithm 26) has two parameters: S_{fvd} and S_{rev}, which are, respectively, two sets of target sequences, one for forward and the other for reverse primers. The sets of sequences are provided by the user and represent desired binding regions for the two primers. The algorithm first finds candidates for forward primers and reverse primers separately. Then, from among these candidates, it iterates over all primer pairs to choose primer pairs with the best joint coverage.

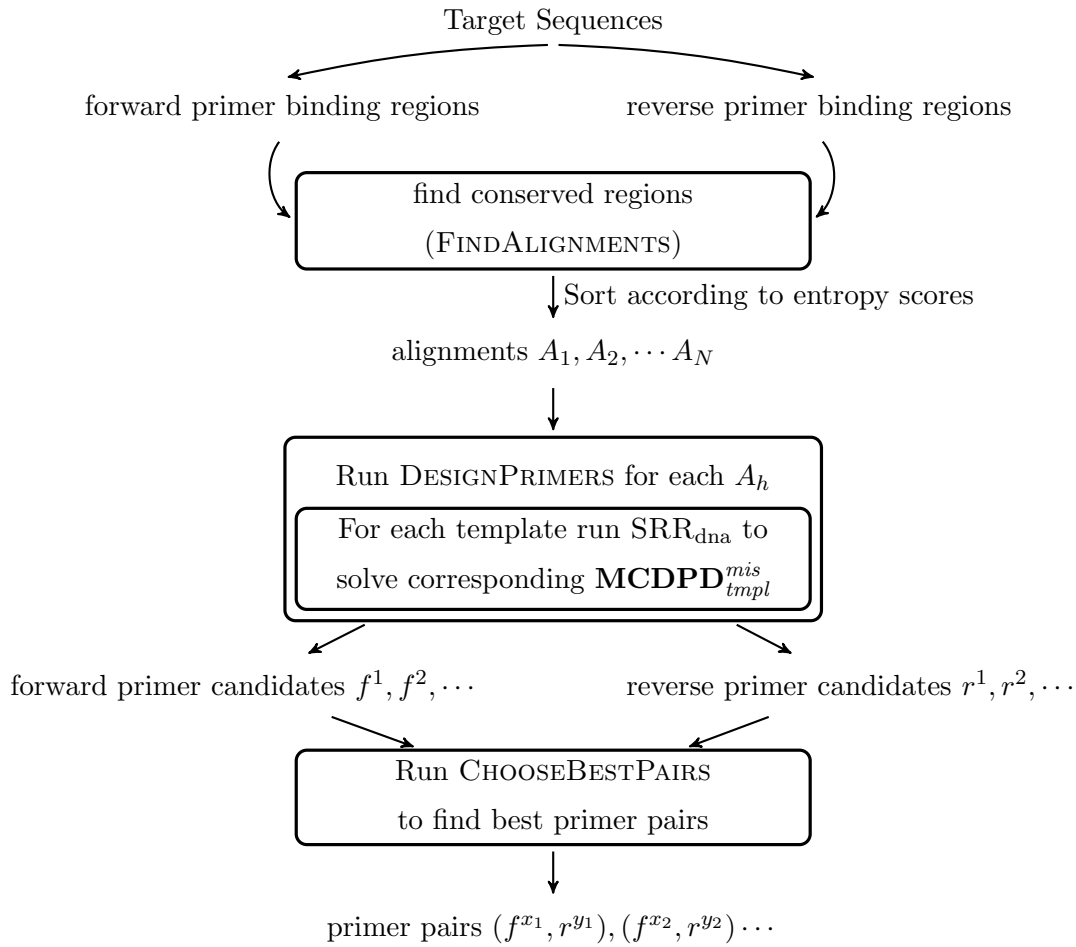


Figure 4.1: Complete Algorithm RRD2P

Algorithm 26 (RRD2P$(S_{fvd} = \{s_f^1, \cdots, s_f^n\}, S_{rev} = \{s_r^1, \cdots, s_r^n\}, k, d, m)$**)**

$PrimerList_{fvd} \leftarrow$ DesignPrimers(S_{fvd}, k, d, λ)

$PrimerList_{rev} \leftarrow$ DesignPrimers(S_{rev}, k, d, λ)

ChooseBestPairs$(PrimerList_{fvd}, PrimerList_{rev})$ ▷ *Find best primer pairs* (f, r)

For both types of primers, the actual primer design process - or, the **Design-Primers** procedure that appears in Algorithm 27 - consists of two parts. In the first part, the algorithm identifies conserved regions within target sequences (Line 1). Much as before, these regions are also called alignments and they are denoted A_h. In the second part, we designed primers for these regions (Lines 2-7).

Algorithm 27 (DesignPrimers$(S = \{s^1, s^2, \cdots, s^n\}, k, d, \lambda)$**)**

1: $A_1, A_2, \cdots A_N \leftarrow$ FINDALIGNMENTS(S, m)
2: ***for all*** *alignments* A_h, $h = 1, \cdots N$ ***do***
3: $\quad PL_h \leftarrow \emptyset$
4: $\quad T_h \leftarrow$ *set of templates* ▷ *see explanation in text*
5: $\quad$ ***for all*** $\hat{p} \in T_h$ ***do***
6: $\quad\quad p \leftarrow \mathrm{SRR}_{dna}(\hat{p}, A_h, d, \lambda)$
7: $\quad\quad$ *Add* p *to* PL_h
8: $PrimerList \leftarrow PL_1 \cup PL_2 \cdots \cup PL_N$
9: ***return*** *PrimerList (sorted according to coverage)*

Finding alignments. Algorithm FINDALIGNMENTS for locating conserved regions (see Algorithm 28) follows the strategy from HYDEN. It enumerates over all sub-strings of length m of the target sequences. For each m-mer, M, we aligned it against every target sequence s^i without gaps to find the best match a^i of length m (i.e., a^i has the smallest Hamming distance with M). The resulting set $A = \{a^1, a^2, \cdots, a^n\}$ of the n best matches, one for each target string, is a conserved region (or, alignment).

Intuitively, more conserved alignments are preferred, since they are more likely to generate low-degeneracy primers. To identify how well-conserved an alignment A is, the entropy score was applied. That is, we sorted alignments according to their entropy scores.

Algorithm 28 (FindAlign($S = \{s^1, s^2 \cdots, s^n\}, m$**))**

1: $AlignmentList \leftarrow \emptyset$
2: ***for all*** m*-mers,* M*, in* S ***do***
3: $\quad A \leftarrow \emptyset$
4: $\quad$ ***for all*** $s^i \in S$ ***do***
5: $\quad\quad a^i \leftarrow$ *substring of* s^i *that is the best match for* M
6: $\quad\quad$ *Add* a^i *to* A
7: $\quad$ *Add* A *to* $AlignmentList$
8: ***return*** *AlignmentList (sorted according to entropy)*

Computing primers. In the second part (Lines 2-7) of Algorithm 27, the algorithm considers all alignments $A_1, ..., A_N$ computed by Algorithm FINDALIGNMENTS. For each A_h, we used the list T_h of template strings (see below). For each $\hat{p} \in T_h$ we call $\mathrm{SRR}_{\mathrm{dna}}(\hat{p}, A_h, d, \lambda)$ that

will compute a primer p, we added them to the list of primers PL_h. All lists PL_h are then combined into the final list of candidate primers. What remains now is an explanation about how to choose the set T_h of templates. If the set $Tmpl_m(A_h)$ of all candidate templates is small, then one can take T_h to be the whole set $Tmpl_\lambda(A_h)$. For instance, when $\lambda = 0$ then $Tmpl_0(A_h) = A_h$. In general, we take T_h to be a random sample of r strings from $Tmpl_m(A_h)$, where the value of r is a parameter of the program that can be used to optimize the tradeoff between the accuracy and the running time. Each $\hat{p} \in T_h$ is constructed as follows: (i) choose uniformly a random $a^i \in A_h$, (ii) choose uniformly a set of exactly m random positions in a^i, and (iii) for each chosen position j in a^i, set a^i_j to a randomly chosen base, where this base is selected with probability proportional to its frequency in position j in all sequences from A_h.

4.4 Results

We tested Algorithm RRD2P on three biological data sets and then compared our results to those from Algorithm HYDEN.

1. The first data set is a set of 50 sequences of human olfactory receptor (OR) gene [41], with a length around 1Kbps, provided via the HYDEN program.

2. The second data set is from the NCBI flu database [1] from which we chose human flu sequences of lengths 900-1000 bps (dated from November 2013). This set contains 229 flu sequences.

3. The third set contains 160 fungal ITS genes of lengths varying from 400-2000 bps that were obtained from NCBI-INSD [2].

For each of these datasets, we ran Algorithm RRD2P with the following parameters:

- Primer length m= 25.

- Primer degeneracy threshold (forward, reverse) : (625,3750), (1250,7500), (1875, 11250), (2500, 15000), (3750, 22500), (5000,30000), (7500,45000), (10000,60000). Note that the degeneracy values increase roughly exponentially, which corresponds to a linear increase in the number of ambiguous bases. We set the degeneracy of the reverse primer to be six times larger than that of the forward primer (the default in HYDEN).

- Forward primer binding range : $0 \sim 300$, reverse primer binding range : $-1 \sim -350$.
- Mismatch allowance : $\lambda = 0, 1, 2$. The value of λ represents the mismatch allowance for each primer separately.
- Number of alignments: $N = 50$.
- Number of template samples: $r = 5$.

For each choice, we compared our algorithm to HYDEN in terms of the coverage of computed primers. To make this comparison meaningful, we designed our algorithm to have similar workflow and used similar input formats and parameters, which allowed us to run HYDEN with exactly the same settings. For the purpose of these experiments, we used the best primer pair from the list computed by Algorithm RRD2P (see Algorithm 26).

For each dataset, the results are shown in Figures 4.2, 4.3 and 4.4, respectively. In each figure, the x-axis represents the degeneracy of the forward primer and the degeneracy of the reverse primer is six times larger. The y-axis is the coverage of the computed primer pair. From these graphs, we see that HYDEN produces some unstable results. Specifically, in some case, higher degeneracy threshold produces lower coverage when other parameters keep unchanged. The results show that RRD2P is capable of finding better degenerate primers than HYDEN for different choices of parameters.

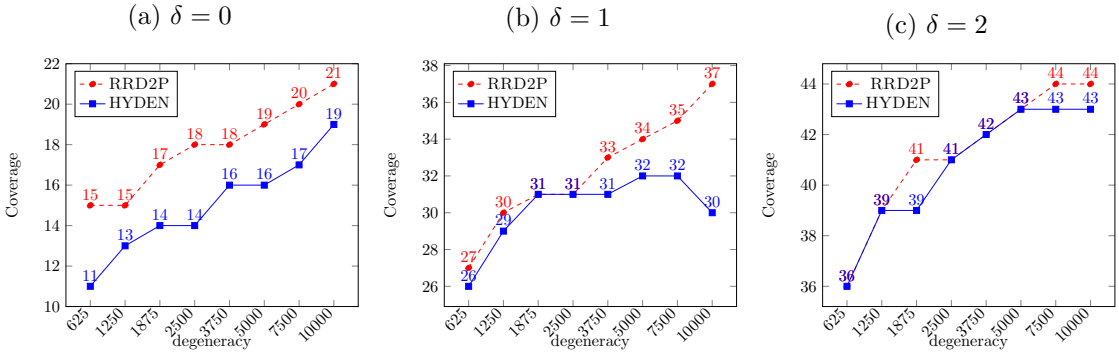


Figure 4.2: Comparison of RRD2P and HYDEN on human OR genes.

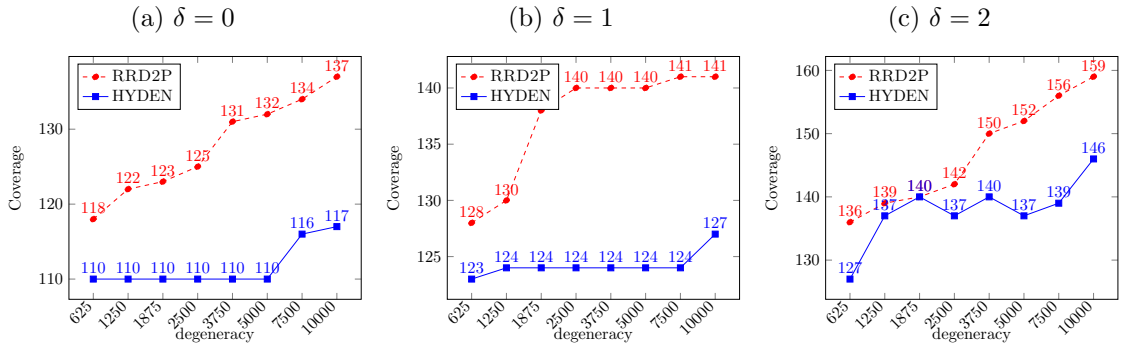


Figure 4.3: Comparison of RRD2P and HYDEN on flu sequences.

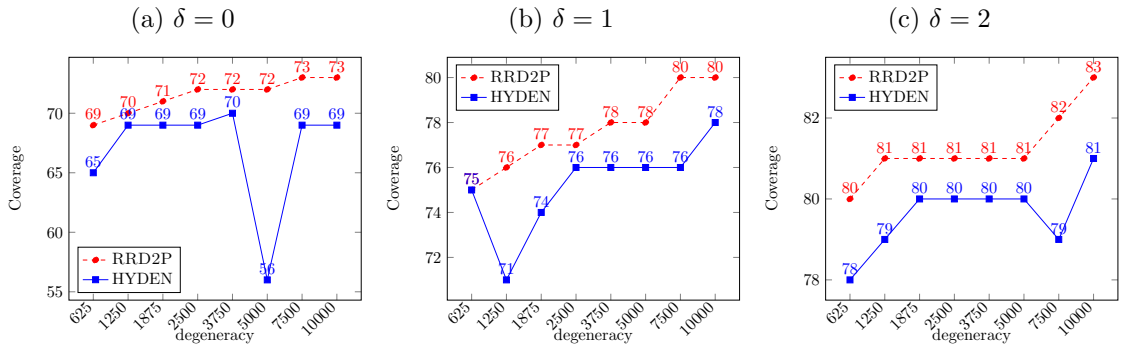


Figure 4.4: Comparison of RRD2P and HYDEN on fungal sequences.

Running time. The running time of Algorithm RRD2P is dominated by the module running Cplex to solve the linear program, which depends, roughly linearly, on the number of times the LP solver runs. The above experiments were performed for $r = 5$. For the third dataset above and $\delta = 0$, the running times of Algorithm RRD2P varied from $110s$ for $d = 625$ to $164s$ for $d = 10000$ (on Windows 8 2.4 GHz CPU, 8.0 G memory). The respective run times of HYDEN were lower: between $25s$ and $28s$. The run time of Algorithm RRD2P can be adjusted by using smaller values of r. For example, for $r = 1, 2$, RRD2P is actually faster than HYDEN for small to moderate degeneracy values and the loss of accuracy is not significant.

Chapter 5

Polynomial Algorithm for the Relaxation of Binary MCDPD$_{tmpl}$

5.1 Linear Program Representation and Dual Problem

In the previous section, the algorithm requires fractional solution of $\mathbf{MCDPD}^{mis}_{tmpl}$ for randomized rounding and solvers such as *CPLEX* are needed. To get rid of solvers, we attempted to develop a heuristic algorithm to compute the fractional solution.

In this section, we propose a heuristic algorithm for solving the restricted case of $\mathbf{MCDPD}_{tmpl}$ that $\lambda = 0$ (problem (22)) with binary alphabet. All strings are binary (i.e., $a^i_j, \hat{p}_j \in \{0, 1\}$ for all i, j, where index j indicates the j_{th} position in a string). We introduce two sets of binary variables:

$$n_j, \quad 1 \leq j \leq m$$
$$x^i, \quad 1 \leq i \leq n.$$

x_j indicates if $p_j = \mathtt{N}$ and y^i indicates if the input string a^i is covered by p. The $\mathbf{MCDPD}_{tmpl}$ can then be represented as a linear program as follows:

$$
\begin{aligned}
\max \quad & \sum_i x^i & & & & \\
\text{subject to} \quad & \sum_j n_j & \leq & \delta & & (5.1) \\
& x^i - n_j & \leq & 0 & \forall i, j : \hat{p}_j \neq a^i_j & (5.2) \\
& n_j, x^i & \in & [0,1] & \forall i, j & (5.3)
\end{aligned}
$$

If there exists some sequence a^i that is equal to the template $\hat{p}$, since the computed primer p covers $\hat{p}$, then a^i is also guaranteed to be covered. Therefore, such sequences do not affect the choice of degenerate primer p and can be ignored in the $\mathbf{MCDPD}_{tmpl}$ problem. After we remove such sequences, each variable y^i must appear in some constraints (5.2).

Therefore, all constraints $x^i \leq 1$ can be eliminated and the constraints (5.3) can be rewritten as

$$
\begin{aligned}
n_j & \leq 1 & \forall j & (5.4) \\
n_j, x^i & \geq 0 & \forall i, j & (5.5)
\end{aligned}
$$

To solve the linear program using combinatorial methods, we attempted to find clues from its dual problem. We introduced dual variables α, β^i_j and γ_j that correspond to constraints (5.1), (5.2) and (5.4), respectively. The dual linear program is given below:

$$
\begin{aligned}
\min \quad & \delta \cdot \alpha + \sum_j \gamma_j & & & & \\
\text{subject to} \quad & \alpha + \gamma_j - \sum_{i:\hat{p}_j \neq a^i_j} \beta^i_j & \geq & 0 & \forall j & (5.6) \\
& \sum_{j:\hat{p}_j \neq a^i_j} \beta^i_j & \geq & 1 & \forall i & (5.7) \\
& \alpha, \beta^i_j, \gamma_j & \geq & 0 & \forall j, i : \hat{p}_j \neq a^i_j & (5.8)
\end{aligned}
$$

5.2 Special case: $\delta = 1$

5.2.1 Linear Program

We first considered the simplified case when $\delta = 1$. In this case, we claim that we can obtain the optimal solution by setting $\gamma_j = 0$ for all j and maximizing α. To justify this

claim, we showed that, given a feasible solution (α, β, γ) which has at least one positive γ_j, we can construct another feasible solution $(\hat{\alpha}, \hat{\beta}, \hat{\gamma})$ with fewer positive γ_j s and with equal or smaller objective value.

Let γ_{min} be the smallest positive γ_j. We can modify the current feasible solution as follows: $\hat{\alpha} = \alpha + \gamma_{min}$, $\hat{\beta} = \beta$, and $\hat{\gamma} = \gamma - \gamma_{min}$. Then, $(\hat{\alpha}, \hat{\beta}, \hat{\gamma})$ is also a feasible solution. In the objective function, the value of α increases by γ_{min} and the value of $\sum_j \gamma_j$ decreases by γ_{min}. Thus, the objective value will not increase.

We can repeat this process until all γ_j are zero, proving our claim. Therefore, when $\delta = 1$, the dual linear program can be simplified as such:

$$
\begin{aligned}
&\min && \alpha && && && \\
&\text{subject to} && \alpha - \sum_{i:\hat{p}_j \neq a_j^i} \beta_j^i && \geq && 0 \qquad \forall j && (5.9)\\
& && \sum_{j:\hat{p}_j \neq a_j^i} \beta_j^i && \geq && 1 \qquad \forall i && (5.10)\\
& && \alpha, \beta_j^i && \geq && 0 \qquad \forall i, j && (5.11)
\end{aligned}
$$

This linear program can be thought of as a flow problem on a bipartite graph $G = (U, V, E)$. Specifically, U has one vertex u_j for each position j, V has one vertex v^i for each sequences a^i, and E is the set of directed edges (v^i, u_j) between any pair of vertices v^i and u_j such that $\hat{p}_j \neq a_j^i$.

In the corresponding flow problem, each vertex $v^i \in V$ is a source into which we inject a flow with value 1. Each edge $e = (v^i, u_j)$ has infinite capacity and some flow β_j^i. (Note: we sometimes use notation $\beta(e)$ or $\beta(i, j)$ instead.) Each vertex $u_j \in U$ is a sink with outflow $\alpha_j = \sum_{i:(i,j)\in E} \beta_j^i$. Then, the objective is to minimize $\alpha = \max(\alpha_1, \cdots \alpha_m)$. Given a feasible solution (α, β), a sink u_j is called a maximizer if $\alpha_j = \alpha$. Typically, we use notation J for the set of maximizers.

The bipartite graph G described above and the flow (α, β) on it is depicted in Figure 5.1.

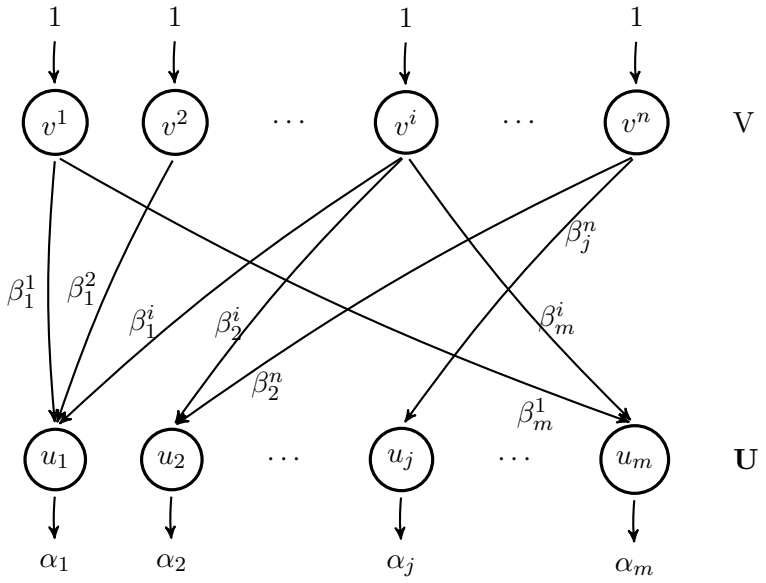


Figure 5.1: Dual Problem

5.2.2 Characterization of Optimal Solutions

Let (α^*, β^*) be the optimal solution of the dual problem and $N(R) \subseteq U$ denote the set of neighbors of a set $R \in V$. We can then prove the following theorem:

Theorem 29

$$\alpha^* = \max_{W \subseteq V} \frac{|R|}{|N(R)|}$$

Proof.

($\geq$) For any subset of sources $R \subseteq V$, the total flow out of R is $|R|$, since each vertex $v \in R$ has outflow 1. This flow can only go to $N(R)$. Therefore, there must exist at least one vertex $u_j \in N(R)$, which has outflow $\alpha_j \geq \frac{|R|}{|N(R)|}$. Hence, the optimal solution α^* will be at least $\frac{|R|}{|N(R)|}$.

($\leq$) Take an optimal solution (α^*, β^*) in which the number of maximizers is minimized (i.e., with minimum $|J|$). Let Q be the set of sources which send flow to J (i.e., $Q = \{v^i : \beta_j^i > 0 \text{ for some } u_j \in J\}$).

According to the definition, flow to J is only contributed by R. Therefore,

$$\alpha^* \leq \frac{|R|}{|J|}.$$

Next, we proved that $N(R) \subseteq J$. Each source $v^i \in R$ must contribute to some sink $u_j \in J$, that is $\beta^i_j > 0$. Suppose that $(v^i, u_k) \in E$ for some $u_k \in U - J$. Using edge (v^i, u_k), we can reduce β^i_j by a small amount $\epsilon > 0$ and increase β^i_k by the same amount ϵ. If $J = \{u_j\}$, this contradicts the optimality of α^*. And, if J contains any sinks other than u_j, then this contradicts the minimality of $|J|$.

$$\alpha^* \leq \frac{|R|}{|N(R)|}$$

■

From the above proof, Lemma 30 below is also true.

Lemma 30 *A feasible solution (α^*, β^*) is optimal if, and only if, there is a non-empty set X of maximizers, such that sources sending positive flow to X do not have any edge to $U - X$.*

5.2.3 Augmenting Paths and Augmentation

Let $p = (u_{t_0}, v^{t_0}, u_{t_1}, v^{t_1}, v^{t_{r-1}}, u_{t_r})$ be an undirected path in G between two sinks u_{t_0} and u_{t_r}. We call (u_{t_a}, v^{t_a}) a backward edge and (v^{t_a}, u_{t_a+1}) a forward edge for all a.

An undirected path p is an *augmenting path* if all its backward edges have positive flow (i.e., $\beta(t_a, t_a) > 0$ for all $a = 0, \cdots, r-1$, as shown in Fig 5.2).

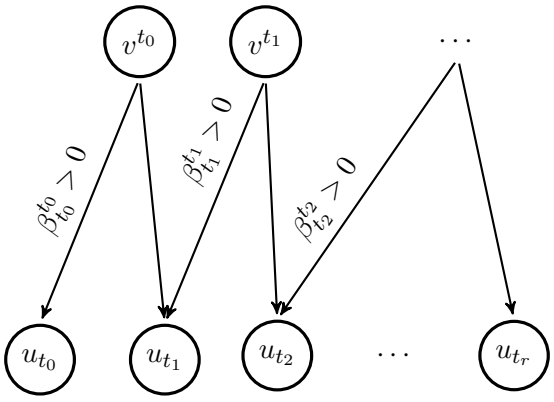


Figure 5.2: Augmenting path

Now we define an *augmentation* of an augmenting path. To augment an augmenting path $p = (u_{t_0}, v^{t_0}, u_{t_1}, v^{t_1}, \cdots, v^{t_{r-1}}, u_{t_r})$ with a certain flow amount $\epsilon > 0$, $\beta(t_a, t_a)$ is reduced by ϵ and $\beta(t_a, t_{a+1})$ is increased by ϵ for all $a = 1, \cdots, r-1$. Since negative flow is not allowed, the value ϵ can be at most $\min_a\{\beta(t_a, t_a)\}$. After the augmentation on p, the updated flow β' still satisfies all linear program constraints and forms a new feasible solution, (α', β').

Procedure *Augment*(J, P, ϵ)**.** Let $J \subseteq U$ be the set of current maximizers for some flow (α, β). From now on, we consider augmenting paths starting in J and ending in $U - J$ only. Given a set of augmenting paths P from J to $U - J$, we can do augmentation on them simultaneously. We define Procedure *Augment*(J, P, ϵ) as augmenting all paths $p \in P$ starting from J with the amount ϵ. We attempt to choose the largest α for which the updated flow (α', β') will remain feasible and such that all maximizers for (α, β) will remain as maximizers for (α', β').

The value of ϵ is bound by two constraints:

1. Since all sinks in J must remain maximizers, the outflow of each $u_k \in U - J$ cannot exceed the new maximum $\alpha - \epsilon$.

2. For edges on which the augmentation decreases flow values, we need to ensure that ϵ is small enough so that their values remain non-negative.

For each $u_k \in U - J$, let $C_P^{end}(u_k)$ be the total number of paths $p_j \in P$ that end at u_k. After we augment P, α_k will increase by $C_P^{end}(u_k) \cdot \epsilon$ (i.e., $\alpha'_k = \alpha_k + C_P^{end}(u_k) \cdot \epsilon$). The new outflow α'_k must be smaller than $\alpha - \epsilon$. Therefore, we need that $\epsilon \leq \epsilon_1(u_k)$, where

$$\epsilon_1(u_k) = \frac{|\alpha - \alpha_k|}{C_P^{end}(u_k) + 1}$$

Now we consider the second constraint. An edge e can be either forward or backward edge in each augmenting path $p \in P$. Let $C_P^{fwd}(e)$ and $C_P^{bkd}(e)$ denote the number of paths $p \in P$ that use e as both the forward edge and backward edge, respectively. After the augmentation, the updated flow will be $\beta'(e) = \beta(e) + \big(-C_P^{bkd}(e) + C_P^{fwd}(e)\big) \cdot \epsilon$. When $C_P^{bkd}(e) \leq C_P^{fwd}(e)$, it is guaranteed that $\beta'(e) \geq \beta(e) \geq 0$. Therefore, we only need to consider edges for which $C_P^{bkd}(e) > C_P^{fwd}(e)$. For those edges, the corresponding bound is

$\epsilon \geq \epsilon_2(e)$, where

$$\epsilon_2(e) = \quad \frac{\beta(e)}{C_P^{bkd}(e) - C_P^{fwd}(e)}.$$

Letting $\epsilon_1 = \min\{\epsilon_1(u_k) \mid u_k \in U - I\}$ and $\epsilon_2 = \min\{\epsilon_2(e) \mid e : C_P^{bkd}(e) > C_P^{fwd}(e)\}$, the largest ϵ for which $Augment(I, P, \epsilon)$, preserves feasibility and all maximizers as

$$\epsilon = \min(\epsilon_1, \epsilon_2). \tag{5.12}$$

If $\epsilon = \epsilon_2(e)$, for some edge e, we call e the bottleneck edge.

5.2.4 Algorithm

The total flow on G is n, since $|V| = n$ and each source v^i has inflow 1. Given an arbitrary feasible solution (α, β) for the dual problem, we can redistribute the flow iteratively to approach the optimal solution. In each iteration, we start from the current feasible solution (α, β), where $\alpha = \max_j \ \alpha_j$.

The idea is to execute Procedure $Augment(J, P, \epsilon)$ in each iteration, doing a simultaneous augmentation by the largest ϵ for which the new solution is feasible and all vertices in J remain maximizers. To achieve polynomial running time, we follow the principle of *Edmonds-Karp* algorithm and choose P to be the set of shortest augmenting paths from each maximizer $u_j \in J$ to $U - J$. The amount ϵ of augmentation is computed using formula (5.12).

Since we choose the shortest paths in P, the bound ϵ_2 can be further simplified. In P, an edge e can be chosen multiple times as either a forward edge or backward edge in different augmenting paths. However, it cannot be forward edge in some path in P and backward in another path in P, or it could make one of these paths shorter. Hence, for each edge e in P, either $C_P^{fwd}(e) = 0$ or $C_P^{bkd}(e) = 0$. The bound ϵ_2 can be then rewritten as

$$\epsilon_2 = \min_{e:C_P^{bkd}(e)>0} \quad \frac{\beta(e)}{C_P^{bkd}(e)}$$

The algorithm for solving the dual linear program is shown in Pseudocode 31.

Algorithm 31 (SolveDual$\big(G=(U,V,E)\big)$) ▷ *Initialization: choose an arbitrary feasible solution* (α,β)

For each v^i*, arbitrarily choose a* u_j *s.t.* $(v^i,u_j)\in E$ *and let* $\beta^i_j \leftarrow 1$

$\beta^i_j = 0$ *for other edges* $(i,j)\in E$

$\alpha_j = \sum_{i:(v^i,u_j)\in E} \beta^i_j \qquad \forall j$

repeat

$\alpha = \max\{\alpha_1,\alpha_2,\cdots,\alpha_m\}$

$J \leftarrow \{u_j | \alpha_j = \alpha\}$

if *Some* $u_j \in J$ *does not have an augmenting path to* $U-J$ ***then***

break

for *each* $u_j \in J$ ***do***

$p_j \leftarrow$ *shortest augmenting path from* u_j *to* $U-J$

$P \leftarrow \{p_j : u_j \in J\}$

$\epsilon_1 \leftarrow \min_{u_k\in U-I} \quad (\alpha-\alpha_k)/\big(C^{end}_P(u_k)+1\big)$

$\epsilon_2 \leftarrow \min_{\{e:C^{bkd}_P(e)\ >0\}} \quad \beta(e)/C^{bkd}_P(e)$

$\epsilon \leftarrow \min(\epsilon_1,\epsilon_2)$ ▷ *If* $\epsilon=\epsilon_1$*, the size of* J *increases, the phase is completed*

▷ *Otherwise, continue another iteration of current phase*

$Augment(J,P,\epsilon)$

end repeat

Correctness. Next, we prove that the obtained solution is optimal. When the algorithm stops, there must exist a non-empty set of maximizers, $X \subseteq J$, which do not have augmenting paths to $U-J$. This implies that X also does not have augmenting paths to $J-X$, otherwise augmenting paths from X to $U-J$ would exist by going through $J-X$. Hence, any sink v^i that contributes to X has no edge to $U-X$. According to Lemma 30, the obtained solution is optimal.

Analysis. In this section, we show that Algorithm *SolveDual* is a polynomial time algorithm. We decided the computation into phases in which each individual phase spans the sequences of augmentations between any two steps when ϵ was chosen to be ϵ_1. In each step

of a phase, except for the last, the value of the augmentation is determined by a bottleneck edge. In the last step, at least one vertex, $u_k \in U - J$, became a maximizer and was added to J. Once a vertex was added into J, which will never be removed. Therefore, the number of phases is at most m. It remains to estimate the number of iterations required to complete a phase. For each vertex $v^i \in V$ and $u_j \in U$, let $d(v^i)$ and $d(u_j)$ be the length of the shortest augmenting path from it to $U - J$; $d(u_k) = 0$ for $u_k \in U - J$. We claim that the distance $d(u_j)$ never decreases. During the augmentation process in which for each edge $e = (v^i, u_j)$, the flow $\beta(e)$ on it may change in three possible ways:

- If $\beta(e) > 0$ increases or decreases but remains positive, this will not affect the shortest augmenting paths from I to $U - J$.
- If $\beta(e) > 0$ is reduced to 0, e can no longer be used as a backward edge in any augmenting path. Removing edges cannot shorten the length of any augmenting path.
- When $\beta(e)$ raises from 0 to some positive value, then $d(v^i) > d(u_j)$ otherwise $\beta(e)$ can not increase. Hence, assume the shortest augmenting path starting from u_j uses e, the length will increase to at least $d(v^i)$. Then, for other $u_k \in U$, if the shortest augmenting path starting from u_k goes through u_j, it must use e as an edge and the distance $d(u_k)$ will also increase. For augmenting paths in which e is not used, the distance keeps the same.

We now claim that between any two times when $e = (v^i, u_j)$ is used as a bottleneck edge, $d(v^i)$ increases by at least 2. Consider a step when e is a bottleneck edge. Then, at this step, e is a backward edge with $\beta(e) > 0$. Letting $r = d(u_j)$, we have $d(v^i) = r - 1$. After the augmentation, $\beta(e)$ is reduced to 0 and cannot become a backward edge; thus, it cannot be a bottleneck edge either. Before e can be a bottleneck edge again, it must have $\beta(e) > 0$, which implies e was a forward edge at some time. At this time, we have $d(v^i) > d(u_j)$ and we also know $d(u_j) = r$ will not decrease, thus $d(v^i) \geq r + 1$. Therefore, $d(v^i)$ increases from $r - 1$ to at least $r + 1$.

Let $|U| = m$, $|V| = n$, and $|E| = b$. The distance between any two nodes in G is at most $m+n$. Therefore, each edge can be used as a bottleneck edge for at most $O(m+n)$ times, according to the earlier claim about the bottleneck edges. Hence, each phase takes at most $O(b(m + n))$ iterations. The maximum number of phases is m and, in each iteration,

it takes time $O(b)$ to search for the shortest augmenting paths. The overall time complexity of the above algorithm is $O(b(m+n)) \cdot O(m) \cdot O(b) = O(b^2 m(m+n))$.

5.3 General Case

5.3.1 Linear Program Interpretation

Recall that the general case dual problem is as below:

$$\min \quad \delta\alpha + \sum_j \gamma_j \tag{5.13}$$

$$\text{subject to} \quad \alpha + \gamma_j - \sum_{i:\hat{p}_j \neq a_j^i} \beta_j^i \geq 0 \qquad \forall j \tag{5.14}$$

$$\sum_{j:\hat{p}_j \neq a_j^i} \beta_j^i \geq 1 \qquad \forall i \tag{5.15}$$

$$\alpha, \beta_j^i, \gamma_j \geq 0 \qquad \forall i, j \tag{5.16}$$

$$\tag{5.17}$$

As in the previously discussed special case, the general case can also be interpreted as a flow problem on a biparite graph. The bipartite graph $G = (U, V, E)$ and the flow on G is constructed in the same way as shown in Figure 5.1: a source vertex $v^i \in V$ for each i that provides flow 1, a sink vertex $u_j \in U$ for each j that outputs flow α_j, and β_j^i indicates the flow on each edge $(v^i, u_j) \in E$.

Since $\alpha_j = \sum_{i:(v^i,u_j)\in E} \beta_j^i$, the constraint (5.14) can be re-written as $\gamma_j \geq \alpha_j - \alpha$. In an optimal solution (α, β, γ), according to the constraints (5.14) and (5.16), we must have $\gamma_j = \max(\alpha_j - \alpha, 0)$. The variable α can be thought of as a threshold and γ_j is the amount that the outflow α_j exceeds this threshold α. We next claim that the minimum objective value equals the sum of δ largest outflows.

Theorem 32 *Assume (α, β, γ) is an optimal solution and the sinks are sorted so that $\alpha_{t_1} \geq \alpha_{t_2} \geq \cdots \geq \alpha_{t_m}$. Then, $\alpha = \alpha_{t_\delta}$ and the objective value (5.13) is equal to $\sum_{j=1}^{\delta} \alpha_{t_j}$.*

Proof. The objective function can be re-written as it is below:

$$\begin{aligned}\delta \cdot \alpha + \sum_{j=1}^{m} \gamma_j &= \sum_{j=1}^{\delta} (\alpha + \gamma_{t_j}) + \sum_{j=\delta+1}^{m} \gamma_{t_j} \\ &\geq \sum_{j=1}^{\delta} \alpha_{t_j} + \sum_{j=\delta+1}^{m} \gamma_{t_j} \geq \sum_{j=1}^{\delta} \alpha_{t_j}\end{aligned}$$

When $\alpha = \alpha_{t_\delta}$, it is trivial that $\alpha_{t_j} = \alpha + \gamma_{t_j}$ for all $0 \leq j \leq \delta$, and $\gamma_{t_j} = 0$ for all $j > \delta + 1$. Therefore, the equality holds and the objective value is $\sum_{j=1}^{\delta} \alpha_{t_j}$. ■

5.3.2 Characterization of Optimal Solutions

Let Q and R be two disjoint subsets of V. We use $\Phi(Q, R)$ to denote the property that Q and R satisfies the following conditions:

1. $N(Q) \cap N(R) = \emptyset$
2. $|N(Q)| < \delta \leq |N(Q)| + |N(R)|$

We now give a characterization of optimal solutions that generalizes Theorem 29. Notice how sets Q and R that satisfy $\Phi(Q, R)$ always exist, since Q can be an empty set, and then R can be any set with $|N(R)| \geq \delta$.

Theorem 33 *Let $(\alpha^*, \beta^*, \gamma^*)$ be an optimal solution of a given instance. Then,*

$$\delta \cdot \alpha^* + \sum_{j=1}^{m} \gamma_j^* = \max_{Q,R \,:\, \Phi(Q,R)} \left(|Q| + \Big(\delta - |N(Q)|\Big) \cdot \frac{|R|}{|N(R)|} \right)$$

Proof. ($\geq$) Fix any feasible solution (α, β, γ), and the sets $QandR$ satisfy the above constraints. Let $I = N(Q)$ and $J = N(R)$. We prove that the following inequality holds:

$$\delta \cdot \alpha + \sum_{j=1}^{m} \gamma_j \;\geq |Q| + (\delta - |I|) \cdot \frac{|R|}{|J|}$$

- case 1: $\alpha \leq \dfrac{|R|}{|J|}$

$$\begin{aligned}
\delta \cdot \alpha + \sum_{j=1}^{m} \gamma_j &\geq \delta \cdot \alpha + \sum_{j:u_j \in I \cup J} \gamma_j \\
&\geq \delta \cdot \alpha + \sum_{j:u_j \in I \cup J} (\alpha_j - \alpha) \\
&= \delta \cdot \alpha + \sum_{j:u_j \in I \cup J} \alpha_j - (|I| + |J|) \cdot \alpha \\
&= \delta \cdot \alpha + |Q| + |R| - (|I| + |J|) \cdot \alpha \\
&= |Q| + |R| - (|I| + |J| - \delta) \cdot \alpha \\
&\geq |Q| + |R| - (|I| + |J| - \delta) \cdot \frac{|R|}{|J|} \\
&= |Q| + (|I| - \delta) \cdot \frac{|R|}{|J|}
\end{aligned}$$

- case 2: $\alpha > \dfrac{|R|}{|J|}$

$$\begin{aligned}
\delta \cdot \alpha + \sum_{j=1}^{m} \gamma_j &\geq \delta \cdot \alpha + \sum_{j:u_j \in I} \gamma_j \\
&\geq \delta \cdot \alpha + \sum_{j:u_j \in I} (\alpha_j - \alpha) \\
&= \delta \cdot \alpha + \sum_{j:u_j \in I} \alpha_j - |I| \cdot \alpha \\
&= \delta \cdot \alpha + |Q| - |I| \cdot \alpha \\
&= |Q| + (\delta - |I|) \cdot \alpha \\
&> |Q| + (\delta - |I|) \cdot \frac{|R|}{|J|}
\end{aligned}$$

($\leq$) Let $(\alpha^*, \beta^*, \gamma^*)$ be the optimal solution with the minimum lexical order. That is, given the sorted outflow $(\alpha^*_{t_1} \cdots \alpha^*_{t_m})$, for the sorted outflow $(\alpha_{t_1} \cdots \alpha_{t_m})$ of any feasible solution (α, β, γ), if $\alpha^*_{t_x} < \alpha_{t_x}$ for some x then there must exist some $\alpha^*_{t_i} > \alpha_{t_i}$ where $1 \leq i < x$.

Let I be the set of sinks u_j where $\alpha_j > \alpha^*$ and J be the set of sinks u_j where $\alpha_j = \alpha^*$. It's trivial that I and J are disjoint sets. Notice that the size I and J must satisfy $|I| < \delta \leq |I| + |J|$, otherwise $(\alpha^*, \beta^*, \gamma^*)$ is not opimal and will form a contradiction. If $|I| \geq \delta$, we can raise α^* by some small amount ϵ to obtain a smaller object value.

Let Q and R be the subsets of V, which contributes positive flow to I and J, respectively. We prove that Q and R can be disjoint. If Q and R are not disjoint, it means there exists some v^i, which contributes to both $u_{t_j} \in I$ and some $u_{t_k} \in J$. We can redistribute the outflow of v^i by sending less to u_{t_j} and sending more to u_{t_k} without changing the order (which has smaller lexical order). This forms a contradiction of minimum lexical order. Therefore, Q and R are disjoint.

It is trivial that Q does not contribute to any $U - (I \cup J)$, otherwise the optimality is violated. Thus, $\sum_{j:u_j \in I} \alpha_j = |Q|$. Also, according to the definition, all flow to J is contributed by R only and each $u_j \in J$ outputs α^*. Hence, $\alpha^* \leq \frac{|R|}{|J|}$.

From the information in the above paragraph, the object value is bounded as below:

$$\begin{aligned}
\delta \cdot \alpha^* + \sum_{j=1}^{m} \gamma^*_j &= \delta \cdot \alpha^* + \sum_{j:u_j \in I} \gamma^*_j \\
&= \delta \cdot \alpha^* + \sum_{j:u_j \in I} (\alpha_j - \alpha^*) \\
&= \delta \cdot \alpha^* + \sum_{j:u_j \in I} \alpha_j - |I| \cdot \alpha^* \\
&= \delta \cdot \alpha^* + |Q| - |I| \cdot \alpha^* \\
&= |Q| + (\delta - |I|) \cdot \alpha^* \\
&\leq |Q| + (\delta - |I|) \cdot \frac{|J|}{|R|}
\end{aligned}$$

■

Similarly, from the above proof, Lemma 34 below is also true.

Lemma 34 *A feasible solution (α, β, γ) is optimal if the set of sinks I which have outflow greater than α do not have common neighbors with $U - I$ and the set of sinks J that have outflow α do not have common neighbors with $U - J$.*

5.3.3 Algorithm

To solve the general case, we can apply Algorithm *SolveDual* repeatedly. Each time, a set of vertices X is identified - vertices that have current maximum outflow and do not have augmenting paths to $U - X$. We call this a stage. Then, we repeat the same process on $U - X$, until all vertices in U is in some set X. This process separates U into groups. Vertices in one group has the same outflow as another and do not have augmenting paths to other groups. The algorithm for solving the generalized dual linear program is shown in Pseudocode 35.

Algorithm 35 (SolveDual2$\big(G=(U,V,E)\big)$)

$I \leftarrow \emptyset$

$\triangleright$ *Initialization: choose an arbitrary feasible solution* (α, β)

For each v^i, *arbitrarily choose a* u_j *s.t.* $(v^i, u_j) \in E$ *and let* $\beta^i_j \leftarrow 1$

$\beta^i_j = 0$ *for other edges* $(i,j) \in E$

while $I \neq U$ ***do***

repeat

$\alpha_j = \sum_{i:(v^i,u_j)\in E} \beta^i_j \qquad \forall j$

$\alpha = \max\{\alpha_j : \alpha_j \in U - I\}$

$J \leftarrow \{u_j | \alpha_j = \alpha\}$ $\triangleright$ $J \subseteq \hat{U}$ *: the set of maximizers in current solution*

if *Some* $u_j \in J$ *does not have an augmenting path to* $U - (I \cup J)$ ***then***

break

for *each* $u_j \in J$ ***do***

$p_j \leftarrow$ *shortest augmenting path from* u_j *to* $U - (I \cup J)$

$P \leftarrow \{p_j : u_j \in J\}$

$\epsilon_1 \leftarrow \min_{u_k \in U-I} \quad (\alpha - \alpha_k)/\big(C^{end}_P(u_k) + 1\big)$

$\epsilon_2 \leftarrow \min_{\{e: C^{bkd}_P(e) \ >0\}} \quad \beta(e)/C^{bkd}_P(e)$

$\epsilon \leftarrow \min(\epsilon_1, \epsilon_2)$ $\triangleright$ *if* $\epsilon = \epsilon_1$, *end phase*

$Augment(J, P, \epsilon)$

end repeat

$X \leftarrow \{u : u \in J$ *which do not have augmenting path to* $U - (I \cup J)\}$

$I \leftarrow I \cup X$ $\triangleright$ *end stage*

Correctness. When Algorithm *SolveDual*2 stops, let I be the set of sinks with outflow greater than the threshold. Sinks $u_j \in I$ do not have any augmenting paths to $U - I$, otherwise its outflow can be further reduced. We did not finish identifying groups and yet the algorithm will not stop. Thus, all sources Q which send positive flow to I do have any edge to $U - I$. Similarly, let J be the set of sinks with outflow equal to the threshold in which J is a group and does not have augmenting paths to $U - J$, otherwise the algorithm

will not stop. Therefore, any sink v^i which contributes to J has no edge to $U - J$. According to Lemma 34, the obtained solution is optimal.

Analysis. Given an instance of $\mathbf{MCDPD}_{tmpl}$ with n input sequences of length m and a template $\hat{p}$, let b be the number of pairs (i, j) such that $\hat{p}_j \neq a^i_j$. If a sequence a^i has more than δ mismatches with the template $\hat{p}$, it is impossible for any primer to cover both a^i and $\hat{p}$. Therefore, we can assume that $b \leq \delta \cdot n$. The constructed bipartite graph $G = (U, V, E)$ in the dual problem has $|U| = m$, $|V| = n$ and $|E| = b$.

Each stage can be finished in polynomial time $b^2m(m+n)$, since it simply applies Algorithm *SolveDual*, as described in the special case $\delta = 1$, to identify a group. Since $b = O(\delta \cdot n)$, the running time for a stage is $O(\delta^2 mn^2(m+n))$.

In the general case, we actually only care about the first δ largest outflows and we do not need to group the whole set of U. In fact, Algorithm *SolveDual*2 can terminate earlier when $|I| \geq \delta$. We only need to identify at most δ groups. Hence, the total time to solve the generalized linear program is $O(\delta^3 mn^2(m+n))$.

Bibliography

[1] Degenerate primer design using computational tools. `http://www.ncbi.nlm.nih.gov/genomes/FLU/FLU.html`. Accessed: 2011.

[2] Degenerate primer design using computational tools. `http://www.ncbi.nlm.nih.gov/genbank/collab/`. Accessed: 2011.

[3] Degenerate primer design using computational tools. `http://biochem218.stanford.edu/Projects\%202011/Brand\%202011.pdf`. Accessed: 2011.

[4] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215(3):403–410, Oct 1990.

[5] C. Arnold and I. J. Hodgson. Vectorette PCR: a novel approach to genomic walking. *PCR Methods Appl.*, 1(1):39–42, Aug 1991.

[6] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *IEEE Symposium on Foundations of Computer Science*, (34):724–733, 1992.

[7] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *IEEE Symposium on Foundations of Computer Science*, (34):323–325, 1992.

[8] S. Arvidsson, M. Kwasniewski, D. M. Riano-Pachon, and B. Mueller-Roeber. QuantPrime–a flexible tool for reliable high-throughput primer design for quantitative PCR. *BMC Bioinformatics*, 9:465, 2008.

[9] K. E. Ashelford, A. J. Weightman, and J. C. Fry. PRIMROSE: a computer program for generating and estimating the phylogenetic range of 16S rRNA oligonucleotide probes and primers in conjunction with the RDP-II database. *Nucleic Acids Res.*, 30(15):3481–3489, Aug 2002.

[10] S. Ayyadevara, J. J. Thaden, and R. J. Shmookler Reis. Anchor polymerase chain reaction display: a high-throughput method to resolve, score, and isolate dimorphic genetic markers based on interspersed repetitive DNA elements. *Anal. Biochem.*, 284(1):19–28, Aug 2000.

[11] S. Ayyadevara, J. J. Thaden, and R. J. Shmookler Reis. Discrimination of primer 3'-nucleotide mismatch by taq DNA polymerase during polymerase chain reaction. *Anal. Biochem.*, 284(1):11–18, Aug 2000.

[12] K. C. Bader, C. Grothoff, and H. Meier. Comprehensive and relaxed search for oligonucleotide signatures in hierarchically clustered sequence datasets. *Bioinformatics*, 27(11):1546–1554, Jun 2011.

[13] S. Balla and S. Rajasekaran. An efficient algorithm for minimum degeneracy primer selection. *IEEE Trans Nanobioscience*, 6(1):12–17, Mar 2007.

[14] Sudha Balla, Sanguthevar Rajasekaran, and Ion M. Greedy heuristics for degenerate primer selection, 2008.

[15] SUDHA BALLA, SANGUTHEVAR RAJASEKARAN, and ION I. MANDOIU. Efficient algorithms for degenerate primer search. *International Journal of Foundations of Computer Science*, 18(04):899–910, 2007.

[16] E. Bent, A. Loffredo, M. V. McKenry, J. O. Becker, and J. Borneman. Detection and Investigation of Soil Biological Activity against Meloidogyne incognita. *J. Nematol.*, 40(2):109–118, Jun 2008.

[17] E. Bent, A. Loffredo, J. I. Yang, M. V. McKenry, J. O. Becker, and J. Borneman. Investigations into peach seedling stunting caused by a replant soil. *FEMS Microbiol. Ecol.*, 68(2):192–200, May 2009.

[18] R. Boyce, P. Chilana, and T. M. Rose. iCODEHOP: a new interactive program for designing COnsensus-DEgenerate Hybrid Oligonucleotide Primers from multiply aligned protein sequences. *Nucleic Acids Res.*, 37(Web Server issue):W222–228, Jul 2009.

[19] J. S. Chamberlain, R. A. Gibbs, J. E. Ranier, P. N. Nguyen, and C. T. Caskey. Deletion screening of the Duchenne muscular dystrophy locus via multiplex DNA amplification. *Nucleic Acids Res.*, 16(23):11141–11156, Dec 1988.

[20] R. Clifford and A. Popa. Maximum subset intersection. *Inf. Process. Lett. 111*, pages 323–325, 2010.

[21] M. Dawande, P. Keskinocak, J.M. Swaminathan, and S. Tayur. On bipartite and multipartite clique problems. *J. Algorithms*, 41(2):388–403, 2001.

[22] J. Duitama, D. M. Kumar, E. Hemphill, M. Khan, I. I. Mandoiu, and C. E. Nelson. PrimerHunter: a primer design tool for PCR-based virus subtype identification. *Nucleic Acids Res.*, 37(8):2483–2492, May 2009.

[23] T. Ehlen and L. Dubeau. Detection of ras point mutations by polymerase chain reaction using mutation-specific, inosine-containing oligonucleotide primers. *Biochem. Biophys. Res. Commun.*, 160(2):441–447, Apr 1989.

[24] J.P. Fitch, S.N. Gardner, T.A. Kuczmarski, S. Kurtz, R. Myers, L.L. Ott, T.R. Slezak, E.A. Vitalis, A.T. Zemla, and P.M. McCready. Rapid development of nucleic acid diagnostics. *Proceedings of the IEEE*, 90(11):1708–1721, Nov 2002.

[25] Q. Fu, P. Ruegger, E. Bent, M. Chrobak, and J. Borneman. PRISE (PRImer SElector): software for designing sequence-selective PCR primers. *J. Microbiol. Methods*, 72(3):263–267, Mar 2008.

[26] J. D. Gans, J. Dunbar, S. A. Eichorst, L. V. Gallegos-Graves, M. Wolinsky, and C. R. Kuske. A robust PCR primer design platform applied to the detection of Acidobacteria Group 1 in soil. *Nucleic Acids Res.*, 40(12):e96, Jul 2012.

[27] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[28] J. E. Hopcroft and R. M. Karp. An algorithm for maximum matching in bipartite graphs. *Soc. Ind. Appl. Math. J. Computation*, pages 225–231, 1973.

[29] M. M. Huang, N. Arnheim, and M. F. Goodman. Extension of base mispairs by Taq DNA polymerase: implications for single nucleotide discrimination in PCR. *Nucleic Acids Res.*, 20(17):4567–4573, Sep 1992.

[30] Y. T. Huang, J. I. Yang, M. Chrobak, and J. Borneman. PRISE2: software for designing sequence-selective PCR primers and probes. *BMC Bioinformatics*, 15:317, 2014.

[31] D. E. Hunt, V. Klepac-Ceraj, S. G. Acinas, C. Gautier, S. Bertilsson, and M. F. Polz. Evaluation of 23S rRNA PCR primers for use in phylogenetic studies of bacterial diversity. *Appl. Environ. Microbiol.*, 72(3):2221–2225, Mar 2006.

[32] K. Ihrmark, I. T. Bodeker, K. Cruz-Martinez, H. Friberg, A. Kubartova, J. Schenck, Y. Strid, J. Stenlid, M. Brandstrom-Durling, K. E. Clemmensen, and B. D. Lindahl. New primers to amplify the fungal ITS2 region–evaluation by 454-sequencing of artificial and natural communities. *FEMS Microbiol. Ecol.*, 82(3):666–677, Dec 2012.

[33] J. K. Jansson and J. I. Prosser. Quantification of the presence and activity of specific microorganisms in nature. *Mol. Biotechnol.*, 7(2):103–120, Apr 1997.

[34] S. N. Jarman. Amplicon: software for designing PCR primers on aligned DNA sequences. *Bioinformatics*, 20(10):1644–1645, Jul 2004.

[35] Subhash Khot. Ruling out ptas for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM J. Comput*, 36:1025–1071, 2006.

[36] Kishori M Konwar, Ion I Mandoiu, Alexander Russell, and Alexander A Shvartsman. Improved algorithms for multiplex pcr primer set selection with amplification length constraints. In *APBC*, pages 41–50, 2005.

[37] S. Kwok, S. Y. Chang, J. J. Sninsky, and A. Wang. A guide to the design and use of mismatched and degenerate primers. *PCR Methods Appl.*, 3(4):39–47, Feb 1994.

[38] S. Kwok, D. E. Kellogg, N. McKinney, D. Spasic, L. Goda, C. Levenson, and J. J. Sninsky. Effects of primer-template mismatches on the polymerase chain reaction: human immunodeficiency virus type 1 model studies. *Nucleic Acids Res.*, 18(4):999–1005, Feb 1990.

[39] E. L. Lim, A. V. Tomita, W. G. Thilly, and M. F. Polz. Combination of competitive quantitative PCR and constant-denaturant capillary electrophoresis for high-resolution detection and enumeration of microbial cells. *Appl. Environ. Microbiol.*, 67(9):3897–3903, Sep 2001.

[40] C. Linhart and R. Shamir. The degenerate primer design problem. *Bioinformatics*, 18 Suppl 1:S172–181, 2002.

[41] C. Linhart and R. Shamir. The degenerate primer design problem: theory and applications. *J. Comput. Biol.*, 12(4):431–456, May 2005.

[42] L. Lovasz and M. D. Plummer. *Matching Theory*, volume 29 of *Annals of Discrete Mathematics*. North-Holland; Amsterdam, 1986.

[43] W. C. Nichols, J. J. Liepnieks, V. A. McKusick, and M. D. Benson. Direct sequencing of the gene for Maryland/German familial amyloidotic polyneuropathy type II and genotyping by allele-specific enzymatic amplification. *Genomics*, 5(3):535–540, Oct 1989.

[44] H. Ochman, A. S. Gerber, and D. L. Hartl. Genetic applications of an inverse polymerase chain reaction. *Genetics*, 120(3):621–623, Nov 1988.

[45] H. Okayama, D. T. Curiel, M. L. Brantly, M. D. Holmes, and R. G. Crystal. Rapid, nonradioactive detection of mutations in the human genome by allele-specific amplification. *J. Lab. Clin. Med.*, 114(2):105–113, Aug 1989.

[46] L. L. Presley, J. Ye, X. Li, J. Leblanc, Z. Zhang, P. M. Ruegger, J. Allard, D. McGovern, A. Ippoliti, B. Roth, X. Cui, D. R. Jeske, D. Elashoff, L. Goodglick, J. Braun, and J. Borneman. Host-microbe relationships in inflammatory bowel disease detected by bacterial and metaproteomic analysis of the mucosal-luminal interface. *Inflamm. Bowel Dis.*, 18(3):409–417, Mar 2012.

[47] A. J. Scupham, L. L. Presley, B. Wei, E. Bent, N. Griffith, M. McPherson, F. Zhu, O. Oluwadara, N. Rao, J. Braun, and J. Borneman. Abundant and diverse fungal microbiota in the murine intestine. *Appl. Environ. Microbiol.*, 72(1):793–801, Jan 2006.

[48] D. Sharma, S. Balla, S. Rajasekaran, and N. DiGirolamo. Degenerate primer selection algorithms. In *Computational Intelligence in Bioinformatics and Computational Biology, 2009. CIBCB '09. IEEE Symposium on*, pages 155–162, March 2009.

[49] Richard Souvenir, Jeremy Buhler, Gary Stormo, and Weixiong Zhang. Selecting degenerate multiplex pcr primers, 2003.

[50] G. P. Srivastava, J. Guo, H. Shi, and D. Xu. PRIMEGENS-v2: genome-wide primer design for analyzing DNA methylation patterns of CpG islands. *Bioinformatics*, 24(17):1837–1842, Sep 2008.

[51] A. Untergasser, H. Nijveen, X. Rao, T. Bisseling, R. Geurts, and J. A. Leunissen. Primer3Plus, an enhanced web interface to Primer3. *Nucleic Acids Res.*, 35(Web Server issue):W71–74, Jul 2007.

[52] Vijay V. Vazirani. *Approximation algorithms*. Springer-Verlag New York, Inc., 2001.

[53] W. A. Walters, J. G. Caporaso, C. L. Lauber, D. Berg-Lyons, N. Fierer, and R. Knight. PrimerProspector: de novo design and taxonomic analysis of barcoded polymerase chain reaction primers. *Bioinformatics*, 27(8):1159–1161, Apr 2011.

[54] D. Y. Wu, L. Ugozzoli, B. K. Pal, and R. B. Wallace. Allele-specific enzymatic amplification of beta-globin genomic DNA for diagnosis of sickle cell anemia. *Proc. Natl. Acad. Sci. U.S.A.*, 86(8):2757–2760, Apr 1989.

[55] D. Xu, G. Li, L. Wu, J. Zhou, and Y. Xu. PRIMEGENS: robust and efficient design of gene-specific probes for microarray analysis. *Bioinformatics*, 18(11):1432–1437, Nov 2002.

[56] J. I. Yang, S. Benecke, D. R. Jeske, F. S. Rocha, J. Smith Becker, P. Timper, J. Ole Becker, and J. Borneman. Population Dynamics of Dactylella oviparasitica and Heterodera schachtii: Toward a Decision Model for Sugar Beet Planting. *J. Nematol.*, 44(3):237–244, Sep 2012.

[57] J. I. Yang, P. M. Ruegger, M. V. McKenry, J. O. Becker, and J. Borneman. Correlations between root-associated microorganisms and peach replant disease symptoms in a California soil. *PLoS ONE*, 7(10):e46420, 2012.

[58] M. Yannakakis. On a class of totally unimodular matrices. *Mathematics of Operations Research*, (2):280–304, 1985.

[59] J. Ye, G. Coulouris, I. Zaretskaya, I. Cutcutache, S. Rozen, and T. L. Madden. Primer-BLAST: a tool to design target-specific primers for polymerase chain reaction. *BMC Bioinformatics*, 13:134, 2012.

[60] J. Ye, J. W. Lee, L. L. Presley, E. Bent, B. Wei, J. Braun, N. L. Schiller, D. S. Straus, and J. Borneman. Bacteria and bacterial rRNA genes associated with the development of colitis in IL-10(-/-) mice. *Inflamm. Bowel Dis.*, 14(8):1041–1050, Aug 2008.

[61] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pages 681–690. ACM, 2006.