Tel-Aviv University
Raymond and Beverly Sackler
Faculty of Exact Sciences
School of Computer Science

The Degenerate Primer Design Problem

Thesis submitted in partial fulfillment of the requirements for
M.Sc. degree in the School of Computer Science, Tel-Aviv University

by

Chaim Linhart

The research work for this thesis has been carried out at
Tel-Aviv University under the supervision of
Prof. Ron Shamir

November 2002

Acknowledgments

I wish to express my thanks to Prof. Ron Shamir for teaching me practically everything I know in computational biology and complexity, and helping me realize this dissertation. I also wish to acknowledge Rani Elkon and Roded Sharan, with whom I worked on several interesting projects, including this thesis. I would like to thank our collaborators in the DEFOG project: Tania Fuchs, Miriam Khen, Gustavo Glusman, Tzachi Pilpel, Doron Lancet (The Weizmann Institute of Science, Rehovot), Barbora Malecova, Uwe Radelof, John O'Brien, Ralf Herwig, Hans Lehrach (Max-Planck Institute for Molecular Genetics, Berlin), and Dmitry Shmulevich (Tel-Aviv University). I thank David Johnson for his remarks on the Bin Packing problem. I owe special thanks to Prof. Dani Halperin for introducing many interesting challenges to me and teaching me so much. Last but not least, I thank my wife, Nami, for her support and love.

This work was partially supported by GIF grant G–0506–183.0396.

Abstract

A PCR primer sequence is called *degenerate* if some of its positions have several possible bases. The *degeneracy* of the primer is the number of unique sequence combinations it contains. We study the problem of designing a pair of primers with prescribed degeneracy that match a maximum number of given input sequences. Such problems occur when studying a family of genes that is known only in part, or is known in a related species. We prove that various simplified versions of the problem are hard, show the polynomiality of some restricted cases, and develop approximation algorithms for one variant. Based on these algorithms, we implemented a program called HYDEN for designing highly-degenerate primers for a set of genomic sequences. We report on the success of the program in an experimental scheme for identifying all human olfactory receptor (OR) genes. In that project, HYDEN was used to design primers with degeneracies up to 10^{10} that amplified with high specificity many novel genes of that family, tripling the number of OR genes known at the time.

Contents

Chapter 1

Introduction

Polymerase chain reaction, or *PCR*, is a technique which is used to increase the number of copies of a specific region of DNA, so that enough DNA is produced to be adequately tested or sequenced. In order to use PCR, one must already know the exact sequences which lie on either side of the DNA region of interest. These sequences are used to design two synthetic DNA oligonucleotides, or *primers*, one complementary to each strand of the DNA double-helix and lying on opposite sides of the target region. The primers are typically of length 20–30. Each cycle of the reaction consists of three steps. First, the DNA is heated, which causes the double strands to separate. Then, the DNA is cooled in the presence of large excess of the two primers to allow these oligonucleotides to hybridize (bind) to the complementary sequences in the genomic DNA. In the third step, the annealed mixture is incubated with DNA polymerase and an abundance of the four nucleotides (A, C, G, T). The polymerase "reads" the opposing strand's sequence and extends the DNA–primer pair by "hooking" complementary nucleotides to the DNA strand. Thus, the regions downstream from each of the two primers are selectively synthesized, and can serve as DNA templates in the next cycle. Each PCR cycle therefore doubles the amount of DNA in the region between the two primers. After 20 cycles, this DNA fragment is amplified a million fold. A typical PCR experiment involves 30–40 cycles and takes several hours.

A PCR primer sequence is called *degenerate* if some of its positions have several possible bases [23]. For example, in the primer: GG{C,G}A{C,G,T}A, the third position is C or G and the fifth is C, G or T. The *degeneracy* of the primer is the number of unique sequence combinations it contains. For example, the degeneracy of the above primer is 6. Degenerate primers are as easy and cheap to produce as regular unique primers, are useful for amplifying several related genomic sequences, and have been used in various applications. Most extant applications use low degeneracy of up to hundreds. In this thesis we study the problem of designing primers of high degeneracy.

In a typical situation, one has a collection of related target sequences, e.g., DNA sequences of homologous genes, and the goal is to design primers that will match as many of them as possible. A naïve solution would be to align the sequences without gaps, count the number of different nucleotides in each position along the alignment and seek a primer-length window (typically 20–30) where the product of the counts is low. Such solution is insufficient because of gaps, the inappropriate objective function of the alignment, and, most notably, the exceedingly high degeneracy: When degeneracy is too high, unrelated sequences may be amplified as well, losing specificity. We may have to compromise by aiming to match many but not necessarily all the sequences. Our goal here is to develop an ad-hoc method for designing primers that will allow tradeoff between the degeneracy and the coverage (the number of matched input sequences). We call this objective Degenerate Primer Design (DPD).

There are various reasons for studying DPD. For example, we were involved in a project with the groups of H. Lehrach (MPI Berlin) and D. Lancet (Weizmann) for finding new human olfactory receptor (OR) genes. At the outset of the project (which preceded the publication of the human genome), only 127 OR genes were known, and the goal was to selectively amplify additional OR genes using degenerate primers. The rationale was that primers which match many of the known genes, would also amplify many new genes from the same family as well, whose sequences are closely related. Most OR genes contain conserved regions, and so the primers would be designed to match such regions. OR genes contain a single 1000bp coding exon, so amplification can be done on the genomic sequence. In gene families that contain introns, the same technique can be applied to selectively amplify cDNAs. The technique can be applied to various families, and to extracting genes from a particular family in an unsequenced species based on the known sequences of family members in a related species. In cDNA analysis, one can use degenerate primers for amplifying and then measuring frequencies of members of a gene family.

DPD is related to the Primer Selection Problem (PSP) [31], in which the goal is to minimize the number of (non-degenerate) primers required to amplify a set of DNA sequences. Several algorithms have been developed to solve this problem, and some take into account various biological considerations and technical constraints (see, e.g., [8]). However, for large gene families, the number of primers needed to cover a large portion of the genes without losing specificity is rather large. Furthermore, since the primers are not degenerate, they do not amplify many of the unknown genes. Traditionally, degenerate primers were usually designed manually by examining multiple alignments of the target sequences. CODEHOP is a program for designing degenerate primers for multiply-aligned protein sequences [35]. For each given multiple alignment, it constructs a pair of primers. Each primer consists of a degenerate 3' core region, typically with degeneracy at most 128, and a 5' non-degenerate consensus sequence that stabilizes annealing. CODEHOP works well for small sets of proteins, taking into account the codon usage of the target genome, as well as the desired annealing temperature. However, it is inappropriate for constructing primers with very high degeneracy on large sets of long genomic sequences.

Since a degenerate primer can be viewed as a motif, DPD is also related to motif finding. However, there are marked differences: Motif algorithms (e.g., MEME [2], Random Projections [5], CONSENSUS [16], AlignACE [19], Multiprofiler [22], Gibbs Sampler [26], WINNOWER [32]) usually produce a profile matrix or a HMM, with no constraint on the maximum degeneracy. Some combinatorial motif finding algorithms do use consensus with degenerate positions (e.g., ARGO [39]), but their goal is to find a "surprising" motif, i.e., a pattern that is unlikely given the background sequence probabilities. In DPD, on the other hand, the "surprise" in a primer is irrelevant, and we care about degeneracy and coverage instead.

1.1 Summary of Thesis Results

In this work we study the DPD problem from theoretical and practical perspectives. From the theoretical point of view, we categorize several variants of the problem. In one key variant we bound the degeneracy and wish to maximize coverage, and in another we wish to minimize degeneracy while requiring full coverage. We give conditions under which the problem is polynomial, but prove that the two variants above and some others are in general $\mathcal{NP}$-Hard. For the maximum coverage variant, we provide several polynomial approximation algorithms. We then describe a practical program called HYDEN [1] for producing high degeneracy primers. The program is a heuristic that builds on ideas analyzed in the theoretical part. HYDEN was applied in the context of searching for new human OR genes, where it designed primer pairs with degeneracy as high as $1.4 \cdot 10^{10}$, perhaps the highest ever used. Theses primers were both very sensitive, leading to a 3-fold increase in the number of known OR genes, and remarkably specific, amplifying a negligible number of non-OR sequences. In addition to the experimental results, we analyze the performance of the primers on a large test set of OR genes, extracted from the first draft of the human genome [14]. We also report preliminary results of an experiment for deciphering the canine olfactory subgenome.

Parts of this work appeared in [27] and [11].

The thesis is organized as follows: In Chapter 2 we give formal definitions of the problems. Chapter 3 gives hardness results and polynomial algorithms for several problem variants. In Chapter 4 we give approximation algorithms. Chapter 5 describes the HYDEN program, and Chapter 6 presents the actual performance of HYDEN in the OR project. A summary and directions for further research are given in Chapter 7.

Chapter 2

Problem Definition

Given a set of DNA sequences, our goal is to design a pair of degenerate primers, so that the primers match and amplify (in the PCR sense) as many of the input sequences as possible. In order to obtain primers that match a large number of known genes, and thus have a good chance to detect new related ones, one should obviously use highly degenerate primers. On the other hand, in order to reduce the probability of amplifying non-related sequences, the degeneracy must be bounded. The problem we faced can thus be informally described as follows. Given a training set of known genes, design a pair of primers, one for the 5' side and another for the 3' side, so that the primers would amplify many of the genes and would have degeneracy that does not exceed a pre-defined limit. For this definition we assume that amplification of a gene occurs when the two primers match (in terms of ungapped local alignment) corresponding subsequences in the gene. The region between the matched subsequences is then amplified. This version is called the Degenerate Primer Design (DPD) problem (a formal definition is given below).

One can extend the degenerate primer design problem in several ways. First, we may want to design several primer pairs so that together they cover the whole training set, when one pair is not enough. Second, we may allow a small number of mismatches between the primers and each amplified gene, as this usually does not inhibit hybridization. Third, we can set a lower bound on the length of the amplified regions, since analysis of the genes is impossible when the amplified fragments are too short.

The following notation will help us formally define the problems. Let Σ denote a finite fixed alphabet. In the case of DNA sequences, Σ ={A,C,G,T}. A *degenerate string*, or *primer*, is a string P with several possible characters at each position, i.e., $P = p_1 p_2 \ldots p_k$, where $p_i \subseteq \Sigma$, $p_i \neq \emptyset$. k is the *length* of the primer. The number of possible character sets at a single position is $\sigma = 2^{|\Sigma|} - 1$. The *degeneracy* of P is $d(P) = \prod_{i=1}^{k} |p_i|$. For example, the primer P^* ={A}{C,G}{A,C,G,T}{G}{T} is of length 5 and degeneracy $d(P^*) = 8$. At *non-degenerate positions*, i.e., positions that contain a single character, we shall often omit the brackets. We will sometimes use an asterisk to denote a fully degenerate position, i.e., a position that includes all possible characters. Hence, $P^* =$A{C,G}*GT. An alternative way to describe a primer is using the NC-IUB (Nomenclature Committee of the International Union of Biochemistry) *nucleotide code* [30], also termed the IUPAC (International Union of Pure and Applied Chemistry) nucleotide code, shown in Table 2.1. According to this notation, P^* can be written as: ASNGT. Let $\delta(P)$ be the number of degenerate positions in P. Since each degenerate position contains between two and $|\Sigma|$ possible characters, $2^{\delta(P)} \leq d(P) \leq |\Sigma|^{\delta(P)}$, or: $\lceil \log_{|\Sigma|} d(P) \rceil \leq \delta(P) \leq \lfloor \log_2 d(P) \rfloor$.

	M = {A,C}		
A = {A}	R = {A,G}	B = {C,G,T}	
C = {C}	W = {A,T}	D = {A,G,T}	N = {A,C,G,T}
G = {G}	S = {C,G}	H = {A,C,T}	
T = {T}	Y = {C,T}	V = {A,C,G}	
	K = {G,T}		

Table 2.1: NC-IUB nucleotide code. Each of the 15 letters encodes a different subset of the genetic nucleic acids alphabet: {A,C,G,T}.

A primer $P^1 = p^1_1 p^1_2 \ldots p^1_k$ is a *sub-primer* of a primer $P^2 = p^2_1 p^2_2 \ldots p^2_k$ of the same length, if $\forall i, 1 \leq i \leq k,\ p^1_i \subseteq p^2_i$. This relation is denoted $P^1 \subseteq P^2$. Obviously, $d(P^1) \leq d(P^2)$. The *union* of the primers P^1 and P^2, denoted $P^1 \cup P^2$, is P^{12} where $p^{12}_i = p^1_i \cup p^2_i$.

A primer $P = p_1 p_2 \ldots p_k$ *matches* a string $S = s_1 s_2 \ldots s_l,\ s_i \in \Sigma$, if S contains a substring that can be extracted from P by selecting a single character at each position, i.e., $\exists j, 0 \leq j \leq l-k$ s.t. $\forall i, 1 \leq i \leq k,\ s_{j+i} \in p_i$. For example, the primer P^* matches the string TGAGAGTC starting from the third position. A *mismatch* is a position i at which $s_{j+i} \notin p_i$. In actual PCR, a few mismatches usually do not prevent hybridization. Unless stated otherwise, we will not allow mismatches. We are now ready to define several problem variants:

Problem 1 Degenerate Primer Design (DPD)
Given a set of n strings and integers k, d, and m, is there a primer of length k and degeneracy at most d that matches at least m input strings?

Figure 2.1 shows a small instance of DPD and a corresponding solution. We defined DPD as a decision problem, rather than an optimization problem. Ideally, one wishes to optimize each of the parameters k, m and d. Since the value of k is usually predetermined by biological or technical constraints (e.g., in PCR experiments, k is usually between 20 and 30), we shall focus on optimizing either m, the *coverage* of the primer, or d, the primer's degeneracy. As we

The Degenerate Primer Design Problem

Input: $n = 5$, $k = 7$, $d = 12$, $m = 4$ (Σ ={A,C,G,T})

S_1 =TCGGC**TTGCAAG**CGTACT

S_2 =GGC**TTCCAGG**TCTTATAAGTC

S_3 =GCTTCCACGGTGCGAATCAGGGCTG

S_4 =A**TTGCTAG**GTTCAGGTA

S_5 =GCAAGGTATC**TTGCCAG**CTTTGA

Solution: $P =$ TT{C,G}C{A,C,T}{A,G}G

Figure 2.1: Example of DPD: A primer of length 7 and degeneracy 12 that covers 4 of the 5 input strings. Matches between the primer and the strings are marked in bold face. The string S_3 is matched from position 3 with a single mismatch.

will prove later on, these two optimization problems remain difficult to solve even if simplified further. Specifically, when designing a primer that matches as many strings as possible, we shall assume that all input strings are of the same length as the primer. When minimizing the degeneracy of the primer, on the other hand, we will seek a full coverage of the input strings, i.e., $m = n$.

Problem 2 Maximum Coverage DPD (MC-DPD)
Given a set of strings of length k and an integer d, find a primer of length k and degeneracy at most d that matches a maximum number of input strings.

Problem 3 Minimum Degeneracy DPD (MD-DPD)
Given a set of strings and an integer k, find a primer of length k and minimum degeneracy that matches all the input strings.

In our practical application, the MD-DPD approach yielded primers with degeneracies too high for successful experiments. We therefore focused on MC-DPD, and applied it with a variety of degeneracy limits imposed by technical constraints (Chapters 4–6).

We shall now define several generalizations of MC-DPD and MD-DPD. As mentioned earlier, a gene is usually amplified even if there are a few mismatches between the primer and the gene. In fact, mismatches near the 3' extension site, i.e., close to the part of the gene that undergoes amplification, are typically more disruptive than mismatches at the 5' side of the primer [23]. The following problem takes into account errors (mismatches) between the primer and the strings, but ignores their position (i.e., we assume that all mismatches are equally disruptive).

Problem 4 Minimum Degeneracy DPD with Errors (MD-EDPD)
Given a set of n strings and integers k and e, find a primer of length k and minimum degeneracy that matches all the input strings with up to e errors (mismatches).

Under many circumstances, a single primer might not suffice, i.e., provide satisfactory coverage, due to its limited degeneracy and the divergence of the input strings. A natural question is whether one could design several primers that, together, would match all the strings.

Problem 5 Minimum Primers DPD (MP-DPD)
Given a set of n strings of length k and an integer d, find a minimum number of primers of length k and degeneracy at most d, so that each input string is matched by at least one primer.

Finally, we may want to construct a pair (or several pairs) of primers, so that many of the input strings match both primers. In gene terms, we would like to design one primer for the 5' side of the genes and another primer for the 3' side — only genes that match both the 5' (sense) and the 3' (anti-sense) primers are amplified by the PCR procedure. We require that an amplified gene matches the primers at separate positions, so that there is no overlap between the match sites.

Problem 6 Maximum Coverage Degenerate Primer Pair Design (MC-DPD2)
Given a set of n strings and integers k, d, find two primers — P_1, P_2, each one of length k and degeneracy at most d, so that a maximum number of input strings match both primers, and the match site of P_1 occurs in all covered strings to the left of the match site of P_2, without overlap between them.

The above definition of MC-DPD2 does not take into account the positions at which each primer matches each gene. In particular, for an effective PCR we should require that the distance between the 5' primer match site and the 3' primer match site is large enough (i.e., the amplified region of the gene is sufficiently long for biological study). This additional constraint does not always pose a problem, as was the case in our application (see Chapter 6) — if the genes contain well-separated conserved regions, we could simply look for good 5' and 3' primers in different, sufficiently far parts of the genes, and thus ensure that the amplified sequences are long enough.

The real problem of designing degenerate primers combines ingredients from all the aforementioned DPD variants. Namely, given a set of input strings, we would like to construct a small set of degenerate primer pairs, so that each of the strings matches at least one of the primer pairs with only a few mismatches. We can also require that each amplified substring is longer than some specified threshold, and incorporate other factors that influence PCR, such as the positions of the mismatches, GC content, and more [23]. Our theoretical results focus on the simple, restricted DPD variants. As we will see in the next chapter, even those are hard. Our heuristics, though, address most of the realistic issues satisfactorily.

Chapter 3

Complexity

In this chapter we shall discuss the computational complexity of the various variants of DPD we defined earlier. Before we prove the hardness of DPD problems, let's examine cases, for which we can suggest a polynomial solution.

3.1 Polynomial-Time Solutions for Restricted Cases

The DPD problem involves several parameters that influence its hardness. We shall now present polynomial-time algorithms for solving DPD when the primer's length (k), degeneracy (d), or coverage (m) are bounded.

3.1.1 Bounded Length

First, let us suppose that k, the length of the primer, is bounded by a constant. Recall that $\sigma = 2^{|\Sigma|} - 1$ is the number of possible character sets in each position of the primer (σ is constant). A straightforward algorithm that checks all the $|\sigma|^k$ possible primers runs in time $O(kL|\sigma|^k)$, where L is the sum of the lengths of the input strings ($O(kL)$ is the time it takes to check a single primer, i.e., count the number of input strings it matches). This naïve algorithm implies:

Theorem 7 *DPD is polynomial when $k = O(\log L)$.* ■

Note that real values of k are bounded (usually, $20 - 30$), but the obtained time bound is impractical.

3.1.2 Bounded Degeneracy

Suppose we bound the degeneracy d of the primer. For the special case of $d = 1$, the non-degenerate primer that matches the maximum number of input strings is clearly a substring of one of the strings. Therefore, we need to check less than L candidate substrings (a string of length l contains $l - k + 1$ substrings of length k), and choose the best one. More generally, if $d = O(1)$, we could consider all $< L$ substrings and continue in one of two ways. First, we could try to increase the degeneracy of each candidate substring by adding new characters at various positions. There are no more than $\delta = \lfloor \log_2 d \rfloor$ degenerate positions in a primer whose degeneracy is d or less, since each such position at least doubles the total degeneracy. At each degenerate position we could try all σ possible character sets. Thus, there are a total of less than $L\binom{k}{\delta}\sigma^{\delta}$ degenerate primers to check, and the total running time is $O(kL^2\binom{k}{\delta}\sigma^{\delta})$.

A different approach would be to take each non-degenerate candidate and expand it using other substrings. Suppose P^1 is a substring of the input string S^1. P^1 can be viewed as a non-degenerate primer ($d(P^1) = 1$) that matches S^1. Let S^2 be an input string that P^1 does not match, and let P^2 be a substring of S^2. Obviously, $P^1 \neq P^2$. Let $P^{12} = P^1 \cup P^2$. P^{12} is a degenerate primer that matches both S^1 and S^2, and its degeneracy is larger than that of P^1 and P^2, since it strictly contains them. Now, P^{12} can be expanded using a third primer, P^3, which is a substring of an input string that is not matched by P^{12}, and so on. We continue to expand the primer as long as its degeneracy does not exceed d. In each step we consider all substrings of the yet un-matched input strings, and add (in terms of the union operation) each substring to the primer, in its turn. Since the degeneracy of the primer increases in each step by at least 1 (more accurately, by a factor of at least $|\Sigma|/(|\Sigma| - 1)$), the number of steps is no more than d. Therefore, the running time of the algorithm is $O(kLL^d)$. In summary:

Theorem 8 *DPD is polynomial when $d = O(1)$.* ■

In Chapter 4 we shall introduce an efficient approximation algorithm that is a judicious variant of the first approach we have just described — expanding a primer candidate by increasing its degeneracy.

3.1.3 Bounded Coverage

Another simple version of DPD is obtained when the number of strings the primer should match is bounded, i.e., $m = O(1)$. As in the case of limited degeneracy, we could enumerate the m k-long substrings the primer matches. If their union is a primer with degeneracy d or less, then it is a valid solution. This algorithm has running time of $O(kL^m)$. In particular:

Theorem 9 *DPD is polynomial when $m = O(1)$.* ■

3.2 Combining MC-DPD and MD-DPD

In the Maximum Coverage DPD problem, we wish to construct a primer the same length as each of the input strings and degeneracy $\leq d$ that matches a maximum number of input strings (Problem 2). This is actually a simplified version of DPD: In the original problem, the input strings have arbitrary length, whereas in MC-DPD they all have length k, which is also the length of the primer we seek. Another simplified DPD variant we defined is MD-DPD (Minimum Degeneracy DPD), where we search for a primer with minimum degeneracy that matches all the input strings (Problem 3). Here, the extra constraint we impose (with respect to the original DPD) is that we require a full coverage, i.e., $m = n$.

As we shall show below, both MC-DPD and MD-DPD are $\mathcal{NP}$-Hard. One may wonder what happens when we combine the two. In other words, is the DPD problem still difficult to solve when all the input strings are of length k, and we seek a primer with degeneracy at most d that covers them all? The answer is no — a trivial polynomial solution is to simply compute the primer P, which is the union of all the input strings, i.e., prepare the set of characters that appear at each position in the strings, as shown in Figure 3.1. If $d(P) \leq d$, then P is a feasible solution. Otherwise, there is no such solution. Interestingly, this polynomial variant of DPD, which we shall denote FCFL-DPD (Full-Coverage Full-Length DPD), regains its $\mathcal{NP}$-Hardness when we allow one mismatch between the primer and each string (see Section 3.3.3), or when we design several primers instead of just one (see Section 3.3.4).

Theorem 10 *FCFL-DPD is polynomial.*

■

FCFL-DPD

Input: $n = m = 5$, $k = 7$, $d = 50$ ($\Sigma =\{$A,C,G,T$\}$)

$S_1 =$ TCGTACG
$S_2 =$ GCATATG
$S_3 =$ GCGAAGG
$S_4 =$ TCCTAAG
$S_5 =$ GCAAATG

Solution: $P = \{$G,T$\}$C$\{$A,C,G$\}\{$A,T$\}$A$\{$A,C,G,T$\}$G

Figure 3.1: Example of FCFL-DPD with 5 strings of length 7: P is the primer with the minimum degeneracy ($d(P) = 48$) that covers all input strings.

3.3 $\mathcal{NP}$-Completeness of Variants of DPD

We shall now study the more difficult cases of DPD, for which exact polynomial-time solutions are not likely to exist.

3.3.1 Maximum Coverage DPD

Our first hardness proof establishes that MC-DPD is $\mathcal{NP}$-Complete, even for a binary alphabet. Since MC-DPD is a special case of DPD, we conclude that DPD is also $\mathcal{NP}$-Complete.

Theorem 11 *MC-DPD is $\mathcal{NP}$-Complete for $|\Sigma| \geq 2$.*

Proof: Clearly, the decision version of MC-DPD is in $\mathcal{NP}$. We complete the proof by reduction from the *Maximum Clique* (CLIQUE, in short) problem, which is $\mathcal{NP}$-Complete ([21], [13, GT19]). Recall that a clique in a graph is a subset of the vertices, in which every two vertices are adjacent.

CLIQUE: Given a graph $\mathcal{G}= (V, E)$ and a positive integer c, is there a clique of size c in $\mathcal{G}$?

Our reduction is illustrated in Figure 3.2. W.l.o.g. we can assume that $c > 3$. We first set $k = |V|$ (the length of the primer and the input strings), $d = 2^c$ (the degeneracy of the primer), and $m = \binom{c}{2}$ (the required coverage). Next, we build $n = |E|$ strings over the binary alphabet $\Sigma = \{0, 1\}$. For each edge in $\mathcal{G}$, we prepare a binary string of length k with 1's at the positions that correspond to the two ends of the edge. Formally, let $V = \{v_1, v_2, \ldots, v_k\}$, and $e = \{v_i, v_j\} \in E$. The string S_e we construct from e is: $S_e = s_1 s_2 \ldots s_k$, where s_x is '1' if $x \in \{i, j\}$, and '0' otherwise. The reduction is clearly polynomial.

We now prove the correctness of the reduction. Assume there is a clique V' of size c in $\mathcal{G}$ — $V' = \{v_{t_1}, v_{t_2}, \ldots, v_{t_c}\}$. Let us examine the primer P that contains degeneracies at the positions that correspond to the c vertices of the clique and 0's at the rest of the positions:

$$P = p_1 p_2 \ldots p_k, \quad p_i = \begin{cases} \{0, 1\} & i \in \{t_1, t_2, \ldots, t_c\} \\ 0 & otherwise \end{cases}$$

P has c degenerate positions and two possible characters at each such position, so its degeneracy is $d = 2^c$. The primer matches every string that corresponds to an edge in the clique, i.e., if $e = \{i, j\}$ and $i, j \in \{t_1, t_2, \ldots, t_c\}$, then P matches S_e. Since there are $\binom{c}{2}$ edges in the clique, it follows that P matches at least m strings, as required.

Conversely, suppose there is a primer $P = p_1 p_2 \ldots p_k$ with degeneracy $d \leq 2^c$ that matches at least $m = \binom{c}{2}$ of the input strings. Since $|\Sigma| = 2$, it follows that each degenerate position is $\{0, 1\}$, and that $d = 2^\delta$, where $\delta \leq c$ is the number of degenerate positions in P. Denote by f

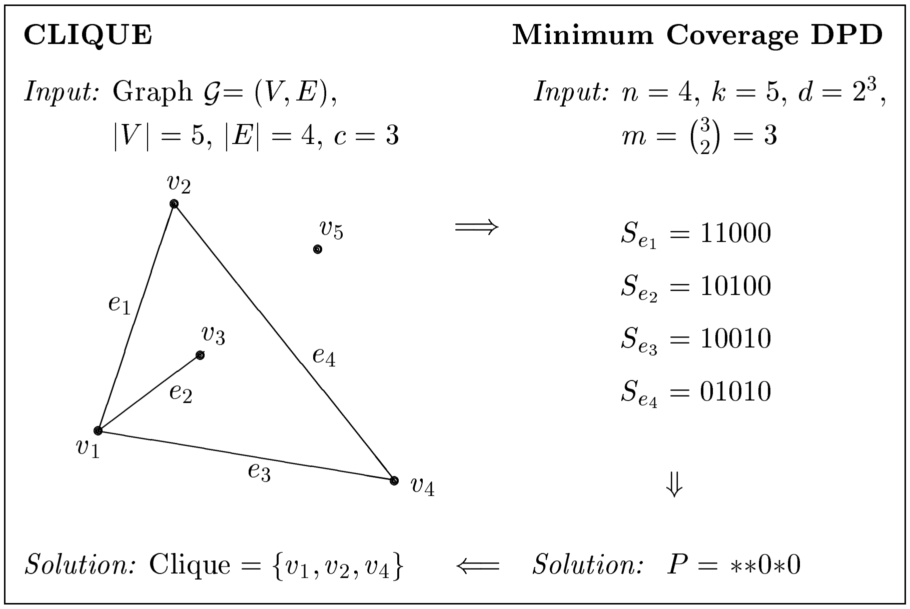

Figure 3.2: Illustration of the reduction from CLIQUE to MC-DPD. The primer P covers the strings S_{e_1}, S_{e_3} and S_{e_4}, which correspond to the edges of the clique. Asterisks in the primer stand for degeneracies ($\{0, 1\}$).

the number of 1's in the non-degenerate positions in P, i.e.: $f = |\{\ i \mid 1 \leq i \leq k,\ p_i = 1\}|$, and let $V' = \{v_{t_1}, v_{t_2}, \ldots, v_{t_\delta}\}$ be the set of vertices that correspond to the degenerate positions, i.e., $p_{t_1} = p_{t_2} = \ldots = p_{t_\delta} = \{0, 1\}$.

Claim 12 *If $c > 3$, then $f = 0$.*

Proof: Notice that all the input strings we constructed contain exactly two 1's. Thus, if $f > 2$, then P does not match any input string, i.e., $m = 0$. Every two vertices in $\mathcal{G}$ are connected by no more than one edge. Hence, if $f = 2$, we get $m \leq 1$ — the primer can only match the string that corresponds to the edge $e = \{v_i, v_j\}$, where i and j are the non-degenerate 1's in P (i.e., $p_i = p_j = 1$). Finally, if $f = 1$, and let $p_i = 1$, then P can only match strings that correspond to edges whose one end is v_i and the other end is in V', and therefore $m \leq |V'| = \delta \leq c$. Thus, we showed that if $f > 0$, it follows that $m \leq c$. On the other hand, $m \geq \binom{c}{2}$, so we get that if $f > 0$, then $\binom{c}{2} \leq c$, which implies that $c \leq 3$, a contradiction. ∎

We now get back to the proof of Theorem 11: According to Claim 12, if $c > 3$, all the non-degenerate positions in the primer P are '0'. Therefore, every input string covered by P contains both its 1's in P's degenerate positions. In other words, the m strings P matches correspond to m edges in the subgraph induced by V'. Since a graph with $|V'| = \delta$ vertices contains no more than $\binom{\delta}{2}$ edges, and since $m \geq \binom{c}{2}$ and $\delta \leq c$, we conclude that $m = \binom{c}{2}$ and $c = \delta$, i.e., V' is a clique of size c, as required. ∎

MC-DPD can easily be reduced to MC-DPD2, by simply concatenating each input string to itself. It is not surprising that designing a pair of primers is at least as difficult as finding a single primer.

Corollary 13 *MC-DPD2 is $\mathcal{NP}$-Complete for $|\Sigma| \geq 2$.* ■

3.3.2 Minimum Degeneracy DPD

Our next result establishes that MD-DPD is $\mathcal{NP}$-Complete, too. It is based on a similar (though simpler) proof in [31] for the problem of finding a minimum *number* of *non-degenerate* primers that cover a given set of sequences.

Theorem 14 *MD-DPD is $\mathcal{NP}$-Complete for $|\Sigma| \geq 3$.*

Proof: The proof utilizes a reduction from another very famous $\mathcal{NP}$-Complete problem — *Minimum Set Cover* (MSC, in short) ([21], [13, SP5]).

MSC: Given a finite set $U = \{e_1, \ldots, e_u\}$, a collection C of subsets of U, and an integer t, are there t members of C whose union is U? (i.e., is there a subset $C' \subseteq C$ such that $|C'| = t$, and every element in U belongs to at least one member of C'?)

Suppose we are given an instance of MSC, and denote: $C = \{C_1, \ldots, C_c\}$. Let $u = |U|$. As in [31], for each element of U we construct an input string for MD-DPD that encodes its subset membership information, i.e., which subsets in C contain it. Let us first define a large alphabet: $\Sigma = \{0, 1, a_1, \ldots, a_c, b_1, \ldots, b_c\}$ (later, we shall show that a ternary alphabet suffices). The characters '0' and '1' will encode the subsets in C, and the a_i's and b_i's will serve as separators, as described henceforth. Define the following strings: $A = a_1a_2 \ldots a_c$ and $B = b_1b_2 \ldots b_c$. We encode each subset $C_j \in C$ by a string G_j of length c that is entirely made up of 0's, except for a single '1' at the j-th position. For example, if $c = 4$, then G_1 is 1000 and G_2 is 0100. We now show how to construct $n = u + 1$ input strings for MD-DPD. Given an element $e_i \in U$ $(1 \leq i \leq u)$, we construct a string S_i, which is a concatenation of strings of the form AG_jB for each subset C_j that contains e_i (w.l.o.g. there is at least one subset C_j that contains e_i, otherwise there is no solution to MSC). Formally, let $C_{j_1}, \ldots, C_{j_x}$ be all the subsets in C that contain e_i, then, $S_i = AG_{j_1}BAG_{j_2}B \ldots AG_{j_x}B$. In addition to the above u strings, we define an auxiliary string $S_0 = AG_0B$, where G_0 is a string with c 0's. Finally, we set $k = 3c$ and $d = 2^t$. Clearly, the reduction is polynomial.

The following simple example clarifies the reduction. Consider an instance of MSC with $U = \{e_1, e_2, e_3, e_4\}$, $C = \{C_1, C_2, C_3\}$, the following subsets: $C_1 = \{e_1, e_2, e_3\}$, $C_2 = \{e_1, e_3\}$ and $C_3 = \{e_1, e_4\}$, and $t = 2$. This instance of MSC has a solution, since $C_1 \cup C_3 = U$.

The reduction constructs five input strings for MD-DPD: $S_0 = A000B$ (the auxiliary string), $S_1 = A100BA010BA001B$ (encoding of e_1), $S_2 = A100B$ (e_2), $S_3 = A100BA010B$ (e_3) and $S_4 = A001B$ (e_4). The separators are: $A = a_1a_2a_3$ and $B = b_1b_2b_3$. The MD-DPD instance asks whether there is a degenerate primer of length $k = 3 \cdot 3 = 9$ and degeneracy at most $d = 2^2 = 4$ that matches all five input strings. The primer $P = A\{0,1\}0\{0,1\}B$ satisfies these conditions. It is easy to observe how P corresponds to the aforementioned solution of MSC.

It is quite clear from our construction how to obtain an adequate primer given a solution to MSC. Suppose $C_{j_1}, \ldots, C_{j_t}$ is a set cover, i.e., $\bigcup_{i=1}^{t} C_{j_i} = U$. As demonstrated in the example, the primer $P = S_0 \cup \bigcup_{i=1}^{t} AG_{j_i}B$ is a valid solution to the MD-DPD instance. The degeneracies in P are all due to the union of G_{j_i}'s. In other words, we could write: $P = Ag_1g_2 \ldots g_cB$, where $g_i = \{0,1\}$ if $i \in \{j_1, \ldots, j_t\}$, and $g_i = 0$ otherwise. Hence, there are exactly t degenerate positions of the form $\{0,1\}$ in P (one per subset in the set cover). Thus, $d(P) = 2^t$, as required. For $1 \le i \le u$, let C_{j_x} be a subset in the set cover that contains e_i. Then, the corresponding string S_i contains the substring $AG_{j_x}B$, which P matches. Since P obviously matches S_0, it follows that it matches all the input strings.

To complete the proof, we now tackle the less obvious task of constructing a solution to MSC given a solution to the corresponding instance of MD-DPD. Let $P = p_1p_2 \ldots p_k$ ($k = 3c$) be a primer of degeneracy d that matches all the input strings (in this reduction $d = 2^t$, but for reasons we shall convey later, we show how to construct a set cover of size $\lfloor \log_2 d \rfloor$ given a primer P of degeneracy d, for *any* d). W.l.o.g. $d \le 2^c$, since the primer $A\{0,1\}\{0,1\} \ldots \{0,1\}B$ (i.e., A followed by c degeneracies of the form $\{0,1\}$, followed by B) matches all the input strings, and its degeneracy is 2^c. $S_0 \subseteq P$, since P matches the auxiliary string S_0, and they have the same length. Consider a match between P and an input string S_i, $1 \le i \le u$. Suppose P matches the substring $M = XAY$ in S_i, where X and Y are substrings, and $|X| > 0$. From the way S_i was constructed, we know that X terminates with the character b_c. Therefore, P contains at least $c+1$ degenerate positions — the positions of the substring b_cA in M:

$$\begin{array}{rcl} S_0 & = & a_1 \ldots\ldots\ a_c\ 0\ 0\ \ldots\ldots \\ M & = & \ldots.\ b_c\ a_1\ \ldots\ldots\ a_c\ \ldots\ldots \end{array}$$

Thus, the degeneracy of P is at least 2^{c+1}, which disagrees with d. We get a similar contradiction if M is of the form YBX, where X is, again, a non-empty substring. The only way P can have degeneracy not larger than 2^c is if it matches a substring $M = AG_{j_i}B$ in S_i (i.e., the match is aligned with the separators). Of course, this must be true for all S_i's, that is, P matches substrings $AG_{j_1}B$,. . . ,$AG_{j_u}B$ in S_1,. . . ,S_u, respectively. Hence, w.l.o.g. all the degeneracies in P are of the form $\{0,1\}$ (other degeneracies do not contribute to the coverage of the primer, so we can omit them), and $d = 2^\delta$, $\delta \le c$. The solution to MSC is clear now — it is the set $C' = \{C_{j_1}, \ldots, C_{j_u}\}$. C' covers U, since for each $1 \le i \le u$, P matches S_i, and so

$e_i \in C_{j_i}$. Finally, $|C'| = \delta$, since each unique member C_{j_x} in C' corresponds to a degeneracy at position j_x in P. Specifically, given that $d(P) = 2^t$, there are exactly t distinct subsets of U in C', as required.

We shall now show that MD-DPD is $\mathcal{NP}$-Complete even if we limit the size of the alphabet to 3. In the above reduction, $|\Sigma| = 2c + 2$, so we can therefore transform any character in Σ to a unique binary string of length $l = \lceil \log_2(2c + 2) \rceil$. Define $\Sigma' = \{0, 1, 2\}$, and transform the strings the reduction constructs to strings over Σ', by replacing each character by $l + 1$ characters as follows. Replace the character '0' by '200...0' (a '2', followed by l 0's), replace '1' by '2100...0' (a '2', followed by a '1', and then $l - 1$ 0's), and replace every other character in Σ by '$2x_1x_2 \ldots x_l$', where '$x_1x_2 \ldots x_l$' is a distinct string of 0's and 1's, and '$x_1x_2 \ldots x_l$' is neither '000...0' nor '100...0'. In this ternary encoding, the input strings are $l + 1$ times longer, and so is the length of the primer we seek: $k' = k(l + 1) = 3c(l + 1)$. Notice that the Hamming distance between the string that represents a '0' and the string that represents a '1' is one. Thus, the degeneracy of the primer we construct in the proof remains unchanged: $d' = d = 2^t$, since all the degeneracies we had were of the type $\{0, 1\}$ (in the ternary encoding, these degeneracies become '2{0,1}00...0'). In addition, a match between a valid primer P' of degeneracy 2^δ ($\delta \leq c$) and an input string must be aligned according to the 2's at the beginning of each character encoding — otherwise, the primer has a degeneracy at least 2^k (the reason is that $S_0 \subseteq P'$, so if the 2's are not aligned, P' contains a degenerate position wherever it matches a '2' in the input string), but $2^k > 2^c$, which is a contradiction. Consequently, the proof of the correctness of the reduction applies for the Σ'-encoded strings, as well. ■

Note that the theorem implies in particular that MD-DPD remains $\mathcal{NP}$-Complete in case of the 4-nucleotides genomic alphabet. Using the reduction above, we can apply a known hardness result of MSC to show that MD-DPD is hard to approximate, as well (though on a different scale). Denote by t_o the value of an optimal solution to an instance of MSC. Given this instance, our reduction builds an instance of MD-DPD with an optimal degeneracy $d_o = 2^{t_o}$. Suppose there exists a polynomial approximation algorithm for MD-DPD that guarantees a solution with degeneracy $d \leq d_o^\alpha$, for some $\alpha \geq 1$. As we have shown in the proof, given a primer of degeneracy d that covers all the input strings, we can construct a solution to MSC of size $\lfloor \log_2 d \rfloor$. Since $d \leq d_o^\alpha = 2^{\alpha t_o}$, we get a set cover of size at most αt_o. However, unless $\mathcal{P} = \mathcal{NP}$, MSC is not approximable in polynomial time within $c \cdot \log u$, for some $c > 0$ [28], and, therefore, $\alpha > c \cdot \log(n - 1)$. In other words, it is difficult to approximate the *number* of degenerate positions, denoted $\delta(P)$, within some logarithmic factor.

Corollary 15 *Assuming $\mathcal{P} \neq \mathcal{NP}$, there exists a constant $c > 0$ such that there is no polynomial-time algorithm for MD-DPD, which is guaranteed to create a solution in which the number of degenerate positions is within a factor of $c \cdot \log n$ of the optimum.* ■

3.3.3 Minimum Degeneracy DPD with Errors

In Section 3.2, we saw that combining MC-DPD and MD-DPD results in a simple polynomial problem, designated FCFL-DPD (Theorem 10). If we generalize this problem by allowing up to one mismatch between the primer and every input string, we get a special case of MD-EDPD, which is $\mathcal{NP}$-Complete, as we shall now prove.

Theorem 16 *MD-EDPD is $\mathcal{NP}$-Complete for $|\Sigma| \geq 2$, even if $e = 1$ and all input strings are of length k.*

Proof: This time, we shall show a reduction from *Minimum Vertex Cover* (MVC, in short) ([21], [13, GT1]).

MVC: Given a graph $\mathcal{G}= (V, E)$ and a positive integer c, is there a vertex cover of size c in $\mathcal{G}$? (i.e., is there $V' \subseteq V$ such that $|V'| = c$, and every edge in E has an endpoint in V'?)

Given an instance of MVC, we construct an instance of MD-EDPD over $\Sigma = \{0, 1\}$ as follows. For each edge $e \in E$, we prepare a binary string S_e of length $|V| + 1$. The string S_e begins with a '0', followed by an encoding of the endpoints of e, as in Section 3.3.1, namely, 1's in the two positions that corresponds to the endpoints, and 0's elsewhere. In addition, we construct an auxiliary string $S_0 = 100\ldots00$ (a '1' followed by $|V|$ 0's). For example, if $|V| = 4$, and $\mathcal{G}$ contains two edges: $e_1 = \{v_1, v_2\}$ and $e_2 = \{v_1, v_4\}$, then we construct three input strings for MD-EDPD: $S_0 = 10000$ (the auxiliary string), $S_1 = 01100$ (encodes e_1), and $S_2 = 01001$ (e_2). Finally, we set $k = |V| + 1$, $d = 2^c$, $m = |E| + 1$ (full coverage), and $e = 1$.

Suppose V' is a vertex cover of size c. Let P be a primer that contains degeneracies at the positions that correspond to the vertices of V', and 0's elsewhere. In our example, $V' = \{v_1\}$ is a vertex cover of size $c = 1$, and $P = 0\{0, 1\}000$ is its corresponding primer. The degeneracy of P is exactly 2^c. For every edge $e \in E$, P matches S_e with one mismatched position (if only one endpoint of e is in V') or with no mismatches at all (if both endpoints are in V'). There is also only one mismatch between P and S_0, occurring at the first character. Thus, P matches each of the m input strings with no more than one mismatch, as required.

Conversely, suppose the primer $P = p_0p_1 \ldots p_{|V|}$ is a solution to MD-EDPD with degeneracy $d = 2^c$. We shall prove that $\mathcal{G}$ has a vertex cover of size c. The following claim is needed for the proof of the theorem.

Claim 17 *If the MD-EDPD instance we constructed has a solution P, then it has a solution P' that begins with a '0' and does not contain 1's, i.e., $P' = 0p'_1 \ldots p'_{|V|}$, and p'_i is either $\{0\}$ or $\{0, 1\}$.*

Proof: To prove the claim, denote by t_i (t'_i) the number of mismatches between P (P') and the i-th input string ($0 \le i \le |E|$). Let us examine p_0, the first position in P. First, if $p_0 = 0$, then P cannot contain 1's — otherwise, it would have at least two mismatches with the auxiliary string S_0, i.e., $t_0 \ge 2$. Thus, in this case we simply set $P' = P$. Second, if $p_0 = 1$, then P may contain another '1' (but not more than one, due to S_0), and $\forall 1 \le i \le |E|$, $t_i = 1$ (a mismatch at the first position). If p_0 is the only '1' in P, then we can set $p'_0 = 0$ and $\forall 1 \le i \le |V|$, $p'_i = p_i$ (i.e., we only change the first position in P from '1' to '0'), and we get: $t'_0 = 1$ (P' has a mismatch with S_0 at the first position), and $\forall 1 \le i \le |E|$, $t'_i = 0$, as required. If, on the other hand, P contains an additional '1', say $p_l = 1$, then we change it to '0', too, and it is easy to see that: $\forall 0 \le i \le |E|$, $t'_i = 1$ (in S_0, the mismatch was moved from position l to position 0; in all other S_i's, the mismatch was moved from position 0 to position l). Finally, if $p_0 = \{0,1\}$, then, as in the previous case, there could be only a single additional '1' in P. Again, we set $p'_0 = 0$ and $\forall 1 \le i \le |V|$, $p'_i = \{0,1\}$ if $p_i = 1$, and, otherwise, $p'_i = p_i$. In other words, we set the first position to '0' (instead of a degeneracy), and we change the single '1', if it exists, to a degeneracy. Notice that $d(P') = d(P)$ or $d(P') = d(P)/2$ (the latter is true if P does not contain 1's). Comparing the number of mismatches of both primers shows that $t'_0 = 1$ and $\forall 1 \le i \le |E|$, $t'_i \le t_i$. In any case, we obtained a primer P' that covers all the strings with up to one mismatch, and whose degeneracy is not higher than that of P. Hence, P' is a valid solution, $p'_0 = 0$ and P' does not contain 1's, as required. ■

We now complete the proof of Theorem 16. According to the claim above, MD-EDPD has a solution P' with degeneracy 2^c, s.t. p_0 is '0' and P' does not contain 1's (that is, P' is entirely made up of 0's and $\{0,1\}$'s). We can now construct a vertex cover of size c for $\mathcal{G}$ by simply choosing all the vertices that correspond to degenerate positions in P'. It is easy to see why this is indeed a vertex cover: for each edge $e \in E$, P' matches S_e with zero or one mismatches, and, therefore, P' contains a degeneracy at least in one of the positions that correspond to the endpoints of e. ■

Using Theorem 16 we can establish an inapproximability result for MD-EDPD, even in the special case of $e = 1$. Håstad has shown that, unless $\mathcal{P} = \mathcal{NP}$, MVC is not approximable within a factor of 1.1666, or more accurately, within $7/6 - \epsilon$, for any $\epsilon > 0$ [18]. Very recently, this hardness bound was improved to 1.36, or $10\sqrt{5} - 21$ [7]. Given a primer with c degenerate positions, our reduction constructs a vertex cover of size c for the corresponding instance of MVC. Hence, we deduce that the number of degeneracies cannot be approximated within a constant factor of 1.36.

Corollary 18 *Assuming $\mathcal{P} \neq \mathcal{NP}$, the number of degenerate positions in MD-EDPD, when we allow one mismatch between the primer and each input string, is not approximable within a factor of* 1.36 *in polynomial time, even when all strings are of length k.* ■

3.3.4 Minimum Primers DPD

In the previous section we studied the complexity of a variant of MD-EDPD, which is a generalization, by allowing mismatches, of FCFL-DPD. Another possible generalization of this problem is the MP-DPD problem, in which we seek several primers, rather than just one primer, that together cover the whole set of input strings. In this section we prove that this problem is $\mathcal{NP}$-Complete.

Theorem 19 *MP-DPD is $\mathcal{NP}$-Complete for $|\Sigma| \geq 2$.*

Proof: Our proof is based on a reduction from *Minimum Bin Packing* (MBP, in short) ([13, SR1]).

MBP: Given l positive integers $a_1, \ldots, a_l$ (the items), and two additional integers c (the capacity) and b (the number of bins), can the items be partitioned into b subsets, each with a total sum of at most c?

MBP is Strongly $\mathcal{NP}$-Complete, i.e., there exists a polynomial p, s.t. MBP remains $\mathcal{NP}$-Complete even if any instance of length l is restricted to contain integers of size at most $p(l)$. We shall assume this restriction in our reduction.

Given an instance of MBP, we construct an instance of MP-DPD over $\Sigma = \{0, 1\}$ as follows. Let $A = \Sigma_{i=1}^{l} a_i$. For each item a_i we prepare a binary string S_i of length A. Let A_i be the sum of the first $i - 1$ items, i.e., $A_i = \Sigma_{i=1}^{i-1} a_i$. The string S_i consists of a prefix of A_i 0's, followed by a_i 1's and a suffix of 0's:

$$S_i = s_1^i s_2^i \ldots s_A^i \ , \qquad s_j^i = \begin{cases} 1 & A_i < j \leq A_i + a_i \\ 0 & otherwise \end{cases}$$

Finally, we set $k = A$, $d = 2^c$, and the target number of primers $p = b$, i.e., we ask whether there are b primers of length A and degeneracy 2^c that match all l input strings. Figure 3.3 illustrates the reduction for a small example. Note that the reduction is polynomial, since all the integers in the input of MBP are bounded by $p(l)$.

Given a solution to MBP — $B_1, \ldots, B_b$, we construct a solution $P_1, \ldots, P_b$ to MP-DPD as follows. Let T_i be the set of positions at which S_i contains 1's, i.e., $T_i = \{A_i + 1, \ldots, A_i + a_i\}$. For bin $B_i = \{a_{i_1}, \ldots, a_{i_u}\}$, we construct the primer P_i that matches the corresponding strings $S_{i_1}, \ldots, S_{i_u}$:

$$P_i = p_1^i p_2^i \ldots p_A^i \ , \qquad p_j^i = \begin{cases} \{0, 1\} & j \in T_{i_1} \cup T_{i_2} \cup \ldots \cup T_{i_u} \\ 0 & otherwise \end{cases}$$

Minimum Bin Packing		**Minimum Primers DPD**
Input: $l = 4$, $c = 5$, $b = 2$		*Input:* $n = 4$, $k = 10$, $d = 2^5$, $p = 2$
$a_1 = 2$		$S_1 = 1100000000$
$a_2 = 1$	$\Longrightarrow$	$S_2 = 0010000000$
$a_3 = 3$		$S_3 = 0001110000$
$a_4 = 4$		$S_4 = 0000001111$
		$\Downarrow$
Solution:		*Solution:*
Bin 1: a_1, a_3	$\Longleftarrow$	$P_1 = {*}{*}0{*}{*}{*}0000$
Bin 2: a_2, a_4		$P_2 = 00{*}000{*}{*}{*}{*}$

Figure 3.3: Illustration of the reduction from MBP to MP-DPD.

The number of degenerate positions in P_i is $|T_{i_1}|+\ldots+|T_{i_u}| = a_{i_1}+\ldots+a_{i_u} \leq c$, as required. Obviously, since every item belongs to one of the bins, every string S_i is covered by one of the primers.

Conversely, let $P_1, \ldots, P_b$ be a solution to MP-DPD. Suppose P_i contains the character '1' at position j, and $j \in T_w$. Then, P_i matches only the string S_w, since all other strings contain a '0' at position j. W.l.o.g., $a_w \leq c$ (otherwise, there is clearly no solution to MBP), so we can replace P_i by a different primer — P_i', which consists of degeneracies at positions T_w, and 0's at the rest of the positions. The degeneracy of P_i' is at most 2^c and it matches S_w, just like P_i. Therefore, we can assume w.l.o.g. that the primers $P_1, \ldots, P_b$ consist only of 0's and degeneracies. It is now clear how to construct a solution for MBP. For each primer P_i we create a bin B_i. If positions T_j are degenerate in the primer P_i, then we add item a_j to bin B_i. The sum of the items we insert into a single bin B_i is at most c, as each degenerate position in P_i contributes at most 1 to this sum. Finally, since each string is covered by at least one primer, it follows that the bins we obtain contain all the given items. ■

Suppose we describe MBP and MP-DPD as optimization functions, rather than decision problems, where the number of bins and the number of primers, respectively, are to be minimized. Then, the above reduction is, in effect, an L-reduction that preserves the target value — a solution with b bins to an instance of MBP is transformed into a solution with b primers to the corresponding instance of MP-DPD, and vice versa. MBP is not poly-time approximable within a factor of $3/2 - \epsilon$ for any $\epsilon > 0$ [13]. Unfortunately, this result does not hold when the input to MBP consists of integers bounded by a fixed polynomial — there are no nontrivial inapproximability results for the strongly $\mathcal{NP}$-Hard version of Bin Packing [20]. Therefore, we cannot apply the L-reduction to prove that MP-DPD is hard to approximate.

As noted earlier, if $p = 1$, MP-DPD becomes FCFL-DPD, which is a polynomial problem (see Section 3.2). For $d = 1$, that is, when no degeneracies are allowed, MP-DPD is the Primer Selection Problem, which is $\mathcal{NP}$-Complete if the input strings are of arbitrary length [31], and polynomial if they are all of length k — the number of primers required is simply the number of unique input strings. Several hardness and inapproximability results for variants of PSP are given in [8].

Chapter 4

Approximation Algorithms

In the previous chapter we proved that many DPD variants, certainly the most interesting ones, are $\mathcal{NP}$-Complete. Thus, unless $\mathcal{P} = \mathcal{NP}$, there are no efficient exact deterministic algorithms for these problems. Even if we apply sophisticated optimization techniques, such as branch and bound, the algorithms will still not have a polynomial running time, and will thus practically be limited for small instances. In order to handle large inputs, we must compromise, either on the exactness of the solution, or on the deterministic nature of the algorithm, or both. We can develop an efficient approximation algorithm that guarantees good, though not optimal, solutions. Hopefully, we could also compute its approximation ratio, i.e., give a lower bound on the quality of the solution with respect to the optimal one. Alternatively, we can employ a randomized algorithm, which gives the optimal solution in high probability. Finally, a common approach is to design heuristics to improve the performance in practice. While lacking theoretical basis, heuristics often yield satisfactory results.

In this chapter we focus on MC-DPD. We developed polynomial approximation algorithms with provable approximation ratios for MC-DPD, when $|\Sigma| = 2$. We implemented a heuristic for the general DPD problem, which is based on our approximation algorithms, and applied it to experimental data (see Chapters 5 and 6). Before exploring the properties of these algorithms, we shall discuss a couple of simple approximation methods. Unless stated otherwise, we shall assume the binary alphabet $\Sigma = \{0, 1\}$, for which the number of degenerate positions in a primer is always $\delta(P) = \log_2 d(P)$. An algorithm is said to yield an approximation ratio r ($r > 1$) if the primer it constructs is guaranteed to match at least m_o/r input strings, where m_o is the coverage of an optimal solution.

4.1 Simple Approximations

Denote by $M(P)$ the set of input strings matched by a primer P. Let P^o be an optimal solution with degeneracy d to an instance of MC-DPD. Like any other primer with degeneracy d, P^o is a union of d non-degenerate primers (strings of length k): $P^o = \bigcup_{i=1}^{d} P^i$, where $P^1,\ldots,P^d$ constitute *all* the non-degenerate sub-primers of P^o, and $M(P^o) = \bigcup_{i=1}^{d} M(P^i)$. Let P^m be a sub-primer with the largest coverage, i.e., $|M(P^m)| = \max_{i=1}^{d}\{|M(P^i)|\}$. Then, obviously, $|M(P^o)| \leq d \cdot |M(P^m)|$. It is now clear how one can obtain a d-approximation to P: Simply traverse all k-long substrings of the input strings, and choose a substring P_0 that matches a maximum number of input strings. Since $|M(P^m)| \leq |M(P_0)|$, we get: $|M(P_0)| \geq |M(P^o)|/d$. The algorithm runs in time $O(kL^2)$, where L is the sum of the lengths of the input strings (in MC-DPD, $L = nk$). The running time can be reduced to $O(kL)$ using a hash table to store the number of strings matched by each substring. Notice that the output of the above algorithm is an optimal non-degenerate primer P_0, and its approximation ratio is d. We can improve the algorithm by finding the optimal primer P_α with α degenerate positions ($1 \leq \alpha \leq \log_2 d$). P_α approximates MC-DPD within a factor of $d/2^\alpha$, since the optimal primer P^o can be represented as a union of $d/2^\alpha$ sub-primers, each one with degeneracy 2^α, s.t. the set of strings covered by P^o is the union of the sets of strings that match the sub-primers. Unfortunately, finding P_α takes exponential time with respect to α.

We now describe another algorithm, which starts with a completely degenerate primer, and gradually "contracts" it. Let P^k be a completely degenerate primer of length k and degeneracy 2^k. P^k covers all the input strings: $|M(P^k)| = n$. We shall now reduce the degeneracy of P^k to d, by replacing $k - \delta$ ($\delta = \log_2 d$) degenerate positions with simple characters. Denote by P_i^k ($i \in \{0,1\}$) the primer that begins with the character i, followed by $k-1$ degeneracies. For example, if $k = 3$, then $P_0^k = 0{*}{*}$ and $P_1^k = 1{*}{*}$. Clearly, $M(P^k) = M(P_0^k) \cup M(P_1^k)$, so by choosing either P_0^k or P_1^k we get a primer whose coverage is at least $n/2$. Similarly, we can de-degenerate, or *refine*, the second position in the primer, i.e., replace it with either '0' or '1', whichever is better, and obtain a primer with degeneracy 2^{k-2} that matches at least $n/4$ input strings, etc. After $k - \delta$ steps we have a primer with the required degeneracy d, whose coverage is at least $n/2^{k-\delta}$, and therefore at least $m_o/2^{k-\delta}$. The total running time of the algorithm is $O((k-\delta)n)$, as it suffices to examine the first $(k-\delta)$ characters in each input string.

Combining the two approximation algorithms we have just described, we can approximate MC-DPD within a factor of $2^{k/2}$: if $\delta < \frac{k}{2}$, we run the first algorithm; otherwise, we execute the second algorithm. In summary:

Proposition 20 *MC-DPD can be approximated within a factor of $2^{k/2}$ in time $O(kL)$.* ■

4.2 Approximating the Number of Unmatched Strings

In this section we describe three approximation algorithms — CONTRACTION, EXPANSION and CONTRACTION-X. Unlike the previous algorithms we studied, these algorithms approximate the number of *unmatched* strings. In other words, instead of expressing MC-DPD as a maximization problem, we now treat it as a minimization problem, designated MC-DPD*, in which the goal is to minimize the number of input strings that the primer does not match, rather than maximizing the number of strings it does match (we now look at the empty half of the glass). This does not alter the optimization problem, only the way in which we measure the quality of the approximation. We say that an algorithm approximates MC-DPD* within ratio r ($r > 1$) if the number of strings not covered by the primer it designs is no more than ru_o, where u_o is the optimal solution value.

The CONTRACTION and EXPANSION algorithms construct the *column distribution matrix* $D(b,i)$ that holds the number of appearances, or *count*, of each character at each position. Formally, denote by $S^j = s^j_1 s^j_2 \ldots s^j_k$ the j-th input string, $1 \le j \le n$, then:

$$\forall\ b \in \Sigma,\ 1 \le i \le k \quad D(b,i) = |\{j \mid s^j_i = b\}|$$

Let $P^o = p^o_1 p^o_2 \ldots p^o_k$ be an optimal primer of degeneracy d, with $\delta = \log_2 d$ degenerate positions. Suppose P^o covers m_o input strings. Denote by u_o the number of strings that P^o does not match, $u_o = n - m_o$. Clearly, $\forall b \notin p^o_i$, $D(b,i) \le u_o$, and for each non-degenerate position i in P^o, $D(p^o_i, i) \ge m_o$. Since P^o contains $k - \delta$ non-degenerate positions, it follows that there are $k - \delta$ (or more) columns in D with a value at least m_o. Given a column distribution matrix D, we define the *leading value* of column i, denoted $v(i)$, as the largest value in that column: $v(i) = \max\{D(b,i) \mid b \in \Sigma\}$. Similarly, the *leading character* of column i is a character $c(i)$, whose count is the leading value: $D(c(i), i) = v(i)$. Let $v(i_1) \ge v(i_2) \ge \ldots \ge v(i_k)$ be the leading values in D, sorted from largest to smallest. The following lemma follows from the discussion above.

Lemma 21 *If P^o covers m_o strings, then $v(i_{k-\delta}) \ge m_o$.* ■

4.2.1 The CONTRACTION Algorithm

The first algorithm we describe is called CONTRACTION. The algorithm selects the $k - \delta$ largest leading values in D, and sets the output primer P^c to contain the $k - \delta$ corresponding leading characters, and degeneracies at the rest of the positions, i.e.:

$$\forall 1 \le i \le k \ \ , \ \ p^c_i = \begin{cases} c(i) & i \in \{i_1, \ldots, i_{k-\delta}\} \\ \{0,1\} & otherwise \end{cases}$$

An alternative way to describe CONTRACTION is as follows. The algorithm starts with a fully degenerate primer, and contracts it iteratively (hence, its name). In each iteration, the algorithm discards the character with the smallest count. In other words, it examines all the remaining degenerate positions, chooses a position i that contains a character b, whose count $D(b, i)$ is smallest, and removes b from position i in the primer. The algorithm stops once the degeneracy of the primer reaches d. In a sense, this is a smart variation of the simple $2^{k-\delta}$-approximation algorithm we saw in the previous section — CONTRACTION uses the column distribution matrix to guide it in selecting good positions to refine, instead of choosing them arbitrarily. Figure 4.1 illustrates an execution of CONTRACTION.

Input: $n = 8$, $k = 9$, $d = 2^4$

S^1: 011010101
S^2: 010010000
S^3: 111010100
S^4: 011111001
S^5: 111010101
S^6: 001111100
S^7: 101011110
S^8: 111010001

$\Longrightarrow$

Column distribution matrix D:

$$\begin{array}{ccccccccc} 4 & 2 & 1 & \mathbf{6} & 0 & 5 & 3 & \mathbf{7} & 4 \\ 4 & \mathbf{6} & \mathbf{7} & 2 & \mathbf{8} & 3 & 5 & 1 & 4 \end{array}$$

$\Downarrow$

Output:

P^c: $*$ 1 1 0 1 $*$ $*$ 0 $*$

Figure 4.1: Example of an execution of CONTRACTION on eight strings. The five ($= k - \delta$) largest leading values in D are marked in bold face. The primer P^c covers four input strings — S^1, S^3, S^5 and S^8.

The running time of CONTRACTION is linear in the length of the input — $O(nk)$, since this is the time it takes to compute the column distribution matrix D, and the $k - \delta$ largest leading values can be found in time $O(k)$ [3, 9]. It remains to prove the approximation ratio. At each degenerate position, the primer P^c has no mismatches with the input strings. Therefore, these positions do not affect the coverage of the primer, and we can ignore them in our analysis. According to Lemma 21, $v(i_1), \ldots, v(i_{k-\delta}) \geq m_o$. Thus, at each non-degenerate position P^c has a mismatch with at most u_o input strings. The total number of strings P^c does not match cannot exceed the sum of the number of mismatches at each position, which is bounded by $(k - \delta)u_o$. In conclusion:

Theorem 22 CONTRACTION *approximates MC-DPD* within a factor of* $(k-\delta)$ *in time* $O(nk)$.

■

4.2.2 The EXPANSION Algorithm

The second algorithm, called EXPANSION, performs n iterations. In each iteration, it expands (degenerates) an input string. In the j-th iteration, EXPANSION computes the matrix D'_j:

$$\forall b \in \{0,1\}\ ,\ 1 \le i \le k\ \ ,\ \ D'_j(b,i) = \begin{cases} 0 & s_i^j = b \\ D(b,i) & otherwise \end{cases}$$

Intuitively, $D'_j(b,i)$ is the number of strings that will be mismatched due to setting the i-th position in the primer to s_i^j while their i-th position is b. EXPANSION then selects the δ largest leading values in D'_j — $v'_j(i_1),\dots,v'_j(i_\delta)$, and uses them to expand S^j and create a primer $P^j = p_1^j \dots p_k^j$, as follows:

$$\forall 1 \le i \le k\ \ ,\ \ p_i^j = \begin{cases} \{0,1\} & i \in \{i_1,\dots,i_\delta\} \\ s_i^j & otherwise \end{cases}$$

The output of the algorithm, P^e, is the best primer P^j it found in the n iterations.

Input: $n = 8$, $k = 9$, $d = 2^4$

S^1: 011010101

S^2: 010010000 $\Longrightarrow$ **Column distribution matrix** D**:**

S^3: 111010100

…(as in Figure 4.1)

$$\begin{array}{ccccccccc} 4 & 2 & 1 & 6 & 0 & 5 & 3 & 7 & 4 \\ 4 & 6 & 7 & 2 & 8 & 3 & 5 & 1 & 4 \end{array}$$

$\Downarrow$

Starting string: S^1 $\Rightarrow$

$$D': \begin{array}{ccccccccc} 0 & 2 & 1 & 0 & 0 & 0 & \mathbf{3} & 0 & \mathbf{4} \\ \mathbf{4} & 0 & 0 & 2 & 0 & \mathbf{3} & 0 & 1 & 0 \end{array}$$

$\Downarrow$

$$P^1: \begin{array}{ccccccccc} * & 1 & 1 & 0 & 1 & * & * & 0 & * \end{array}$$

Starting string: S^2 $\Rightarrow$

$$D': \begin{array}{ccccccccc} 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{4} & 0 & \mathbf{7} & 2 & 0 & 3 & \mathbf{5} & 1 & \mathbf{4} \end{array}$$

$\Downarrow$

$$P^2: \begin{array}{ccccccccc} * & 1 & * & 0 & 1 & 0 & * & 0 & * \end{array}$$

Figure 4.2: Illustration of the first two iterations of EXPANSION on the eight strings from Figure 4.1. The four ($= \delta$) largest leading values in D' are marked in bold face. The expansion of S^1 (P^1) covers four strings, and is identical to the primer constructed by CONTRACTION. The expansion of S^2 (P^2) covers five input strings — S^1, S^2, S^3, S^5, and S^8.

Denote by m_c and m_e the number of strings covered by the primers P^c and P^e, respectively. Lemma 23 establishes that P^e is at least as good as P^c, and, therefore, EXPANSION also guarantees a $(k-\delta)$-approximation to MC-DPD*. In fact, as the lemma implies, in some cases EXPANSION may find a better primer than CONTRACTION, as demonstrated in Figure 4.2. On the down side, EXPANSION is slower — its running time is $O(n^2k)$, dominated by the coverage computation of the n primers it constructs.

Lemma 23 $m_e \geq m_c$.

Proof: Let S^j be a string covered by P^c. We shall prove that EXPANSION expands S^j into P^c, i.e., $P^j = P^c$, which implies $m_e \geq m_c$. Let $v(i_1), \dots, v(i_{k-\delta})$ be the $k-\delta$ largest leading values in D. CONTRACTION sets positions $i_1, \dots, i_{k-\delta}$ in P^c as the corresponding characters in S^j, and the rest δ positions in P^c are degenerate. Since $|\Sigma| = 2$, each column in D has two entries, whose sum is n. Therefore, the complement characters of $c(i_1), \dots, c(i_{k-\delta})$ have the smallest count in D, so the δ largest counts in D'_j cannot be in those columns. In other words, the δ leading values selected in the j-th iteration of EXPANSION are from the columns: $\{1 \leq i \leq k \mid i \neq i_1, \dots, i_{k-\delta}\}$. Thus, P^j is exactly P^c. Note that if different characters have equal counts, the proof does not hold. We can easily fix this, by modifying the sort functions of the algorithms, so that leading values with equal counts are sorted according to their column index in ascending (descending) order in CONTRACTION (EXPANSION). ■

Corollary 24 EXPANSION *approximates MC-DPD* within a factor of* $(k-\delta)$ *in time* $O(n^2k)$.

■

4.2.3 The CONTRACTION-X Algorithm

We now present an improved version of CONTRACTION, called CONTRACTION-X, that yields better approximations at the expense of longer running times. A similar improvement could be developed for the EXPANSION algorithm, as well. The main idea we employ is to examine several positions simultaneously, and decide which are best to refine (i.e., de-degenerate), instead of checking the distribution at each position separately. Formally, let x be a predefined integer, $1 \leq x \leq k-\delta$. For simplicity, assume $x \mid (k-\delta)$. Denote by $\bar{b} = (b_1, \dots, b_x)$ a binary vector of length x, or x-tuple, and denote by $\bar{i} = (i_1, \dots, i_x)$, $1 \leq i_j \leq k$, a set of x distinct positions. Define the *multi-column distribution matrix* $MD(\bar{b}, \bar{i})$ as the count of the x bits of $\bar{b}$ at positions $i_1, \dots, i_x$ in the input strings, i.e.:

$$MD((b_1, \dots, b_x), (i_1, \dots, i_x)) = |\{j \mid s^j_{i_1} = b_1, \ \dots, \ s^j_{i_x} = b_x\}|$$

Let P^o be an optimal primer, and denote by u_o the number of input strings it does not match. CONTRACTION-X starts with a completely degenerate primer, $P^x = p^x_1 \dots p^x_k$, $p^x_j = \{0,1\}$, and iteratively refines it. In the first iteration, it selects an x-tuple with the largest count and sets the x corresponding positions in the primer to contain the bits of the x-tuple. In other words, if $MD(\bar{b}', \bar{i}') = \max\{MD(\bar{b}, \bar{i})\}$, then: $\forall 1 \leq j \leq x$, $p^x_{i'_j} = b'_j$. In the next iteration, CONTRACTION-X continues to refine P^x in a similar fashion. It examines all x-tuples in positions that are still degenerate, i.e., that were not refined in the first iteration, selects an x-tuple with the largest count, and sets the corresponding positions in P^x accordingly. The algorithm performs $\frac{k-\delta}{x}$ iterations, as above, and reports the obtained primer P^x. Since in each iteration it refines x new positions, the output primer contains exactly δ degeneracies, as required. If $x \nmid (k-\delta)$, and denote $r = (k-\delta)$ mod x, then CONTRACTION-X performs $\lfloor \frac{k-\delta}{x} \rfloor$ iterations as above, and an additional iteration, in which it refines only r positions, that is, it computes the count of every r-tuple at each subset of r positions that are still degenerate, selects the largest one, and refines those positions accordingly.

A sample execution of CONTRACTION-X on seven input strings, with $k = 7$, $\delta = 3$ and $x = 2$, is illustrated in Figure 4.3. Notice that for $x = 1$, CONTRACTION-X is identical to CONTRACTION. In the other extreme case, when $x = k - \delta$, CONTRACTION-X effectively considers all k-long primers with δ degeneracies, and it therefore always yields an optimal primer. The multi-column distribution matrix is also utilized in Multiprofiler, a motif finding algorithm that has recently been reported to detect particularly subtle motifs [22].

Theorem 25 CONTRACTION-X *approximates MC-DPD* within a factor of* $\lceil \frac{k-\delta}{x} \rceil$ *in time* $O(\binom{k}{x} n(k-\delta))$ *and space* $O(\binom{k}{x} nx)$.

Proof: Suppose that $x \mid (k-\delta)$. Let us examine the j-th iteration of CONTRACTION-X. At the beginning of the iteration, the primer P^x contains at least $\delta + x$ degenerate positions (actually, it contains exactly $k - (j-1)x$ degeneracies). W.l.o.g., P^o contains exactly δ degeneracies (otherwise, we can add degeneracies to it, without changing its coverage). Thus, there are at least x degenerate positions in P^x that are not degenerate in P^o. Denote them $i_1, \dots, i_x$. P^o does not match u_o input strings, hence:

$$\max\{MD(\bar{b}, \bar{i})\} \geq MD((p^o_{i_1}, \dots, p^o_{i_x}), (i_1, \dots, i_x)) \geq n - u_o$$

Therefore, in each iteration, CONTRACTION-X refines x positions, s.t. the x-tuple it sets at these positions has mismatches with at most u_o input strings. The total number of strings P^x does not match is, in the worst case, the sum of the number of mismatched strings in each iteration, which is at most $\frac{k-\delta}{x} u_o$. If $x \nmid (k-\delta)$, the algorithm performs $\lfloor \frac{k-\delta}{x} \rfloor + 1$ iterations, so the number of strings P^x does not cover is at most $\lceil \frac{k-\delta}{x} \rceil u_o$.

Input: $n = 7$, $k = 7$, $d = 2^3$

S^1: 1100101
S^2: 1100000 $\Longrightarrow$
S^3: 0111000
S^4: 1000100
S^5: 1110111
S^6: 1000001
S^7: 1100100

Bi-column distribution matrix MD:

	1,2	1,3	1,4	...	3,6	...	6,7
0,0	0	0	0	...	**5**	...	4
0,1	1	1	1	...	0	...	2
1,0	2	5	**6**	...	1	...	0
1,1	4	1	0	...	1	...	1

$\Downarrow$ Iteration 1

P^x : $1 \ast \ast \ 0 \ast \ast \ast$

$\Downarrow$ Iteration 2

Output: P^x : $1 \ast 0 \ 0 \ast 0 \ast$

Figure 4.3: Example of an execution of CONTRACTION-X ($x = 2$) on seven strings. The largest bi-column count is $MD((1,0),(1,4)) = 6$, so the first iteration refines positions 1, 4 to '1', '0', respectively. Ignoring positions 1 and 4, the largest remaining count is $MD((0,0),(3,6)) = 5$. Thus, in the second iteration positions 3 and 6 are set to '0'. The output primer covers five input strings — S^1, S^2, S^4, S^6 and S^7.

The matrix MD contains $2^x\binom{k}{x}$ entries, and can be computed in time $O(2^x\binom{k}{x}nx)$. Since MD might be sparse, especially when x is relatively large, a more efficient representation of MD in terms of time, as well as space, is an array A of $\binom{k}{x}$ hash tables — the entry $A(\bar{i})$ in the array contains a hash table with the counts of all x-tuples that appear at positions $\bar{i}$ in the input strings. For each $\bar{i} \subseteq \{1,\ldots,k\}, |\bar{i}| = x$, and for each input string, we add the x-tuple at positions $\bar{i}$ in the string to the hash table $A(\bar{i})$ (with an initial count of 1), or increment the count of the x-tuple if it already exists in $A(\bar{i})$. A contains the count of a total of $O(\binom{k}{x}n)$ x-tuples. The construction of A takes $O(\binom{k}{x}nx)$ time and space. In each iteration of CONTRACTION-X we find a pair $(\bar{b},\bar{i})$ with the maximum count in the sub-matrix of MD induced by the degenerate positions in P^x (i.e., we ignore a column $\bar{i} = (i_1,\ldots,i_x)$ if $\exists j$, s.t. $p^x_{i_j} \neq \{0,1\}$). A single iteration can be performed in time linear in the size of A, or $O(\binom{k}{x}nx)$ — for each of the $O(\binom{k}{x}n)$ entries in A, we check in time $O(x)$ whether its x positions are still degenerate in P^x, and find the largest count among all those entries. The total running time is, thus, $O(\binom{k}{x}n(x + x\lceil\frac{k-\delta}{x}\rceil))$, or $O(\binom{k}{x}n(k-\delta))$.

■

4.2.4 Non-Binary Alphabets

So far, we have discussed several approximation algorithms for MC-DPD when $|\Sigma| = 2$. However, in many real-life applications the alphabet is not binary, as is the case when designing primers for genomic sequences ($|\Sigma| = 4$). The simple approximations described in Section 4.1 are easily generalized to large alphabets, as we shall now show. Let P^o be an optimal primer of length k and degeneracy d for a given set of n strings over Σ. Let m_o be the coverage of P^o. The primer P^o is a union of d non-degenerate primers, and the number of strings covered by P^o is at most the sum of the coverage of these non-degenerate primers. Hence, an optimal non-degenerate primer, which is simply a k-long substring that appears in the largest number of input strings, covers at least m_o/d strings.

As in the binary case, we can also devise a simple contraction algorithm for non-binary alphabets. For convenience, denote $\alpha = |\Sigma|$, and $\delta' = \lfloor \log_\alpha d \rfloor$. A completely degenerate primer of length k has degeneracy α^k and coverage n. By replacing the first degeneracy in the primer with a simple character (one that gives the largest coverage) we get a primer with degeneracy α^{k-1} that covers at least n/α strings. We similarly refine positions $2, \ldots, k - \delta'$, and obtain a primer with degeneracy at most d and whose coverage is at least $n/\alpha^{k-\delta'}$, and therefore at least $m_o/\alpha^{k-\delta'}$.

Both algorithms we have just outlined run in time $O(kL)$, as explained in Section 4.1. Combining them, we get a $|\Sigma|^{\lceil k/2 \rceil}$-approximation algorithm for MC-DPD: if $d \geq |\Sigma|^{\lceil k/2 \rceil}$, then $\alpha^{k-\delta'} \leq |\Sigma|^{\lceil k/2 \rceil}$, so we run the second algorithm; otherwise, we run the first algorithm (compare to Proposition 20). In general:

Proposition 26 *When $|\Sigma| > 2$, MC-DPD can be approximated within a factor of $|\Sigma|^{\lceil k/2 \rceil}$ in time $O(kL)$.*

■

Unfortunately, the results we obtained in Section 4.2 for the CONTRACTION and EXPANSION algorithms do not hold for non-binary alphabets. There are two complications in large alphabets. First, there is more than one possibility for a degenerate position. When $|\Sigma| = 2$, every degenerate position in the primer is $\{0, 1\}$, whereas when $|\Sigma| > 2$ we need to choose one among several possible degeneracies (subsets of Σ with more than one character) at each degenerate position. Second, there is the additional complexity in deciding how to partition the degeneracy between the positions. In the binary case, the degeneracy is always of the form 2^δ, where δ is the number of degenerate positions. However, when $|\Sigma| > 2$, the number of degenerate positions could be any one of many values. For example, if $d = 16$ and $|\Sigma| = 4$, there may be four degenerate positions (each one with degeneracy 2), three $(4, 2, 2)$, or only two $(4, 4)$. In the next chapter, we describe heuristics for MC-DPD with non-binary alphabets that are based on CONTRACTION and EXPANSION, and perform well in practice.

Chapter 5

Implementation: The HYDEN Program

We developed and implemented an efficient heuristic, called HYDEN [1], for designing highly degenerate primers. The input to HYDEN is a list of DNA sequences and a set of integers that specify the length of the primer, its maximum degeneracy, and the number of mismatches it is allowed to have with every sequence it covers. HYDEN constructs a primer with the specified length and degeneracy that covers many of the given sequences. It does so by running a 3-phase algorithm, outlined in Figure 5.1. In the first phase, HYDEN locates conserved regions in the DNA sequences by finding ungapped local alignments with a low entropy score. In the second phase, it designs primers using variants of the CONTRACTION and EXPANSION algorithms. Finally, it uses a greedy hill-climbing procedure to improve the primers, and selects the one with the largest coverage as the output. HYDEN is written in C++, and runs under Windows and Linux.

```
HYDEN ($I = \{S^1, \dots, S^n; k; d; e\}$):
Phase 1: $A_1, \dots, A_{N_a} \leftarrow$ H-Align($I$).
Phase 2: Foreach alignment $A_i$, $i = 1, \dots, N_a$ do:
             $P_i^c \leftarrow$ H-Contraction($I; A_i$).
             $P_i^e \leftarrow$ H-Expansion($I; A_i$).
         Sort primers $\{P_i^c, P_i^e \mid i = 1, \dots, N_a\}$ acc. to coverage.
Phase 3: Foreach primer $P \in$ {best $N_g$ primers} do:
             $P \leftarrow$ H-Greedy($I; P$).
Output the primer with the largest coverage found in Phase 3.
```

Figure 5.1: The HYDEN algorithm.

Formally, let $I = \{S^1, \ldots, S^n; k; d; e\}$ be the input to HYDEN, where $S^1, \ldots, S^n$ are n strings over $\Sigma = \{A,C,G,T\}$ with a total length of L characters, and k, d, and e are the length, degeneracy, and mismatches parameters, respectively. Let N_a, $N_{a'}$, N_g and N_h be additional integer parameters, whose roles will be explained soon. Denote by A an ungapped local alignment (alignment, in short) of the input strings, that is, a set of n substrings of length k (actually, A is a multi-set, since it may contain several copies of a substring). Denote by D_A the column distribution matrix of the substrings in A. In order to determine how well-conserved the alignment is, and thereby estimate how likely we are to construct a good primer from it, we compute its entropy score, H_A:

$$H_A = -\sum_{i=1}^{k} \sum_{b \in \Sigma} \frac{D_A(b,i)}{n} \cdot \log_2 \frac{D_A(b,i)}{n}$$

The lower the entropy score is, the less variable are the columns of A, and, intuitively, the greater the chances are for finding a primer that covers many of the substrings in A. The first phase of HYDEN, called H-ALIGN, exhaustively enumerates all substrings of length k in the input strings, and generates an alignment for each one, as follows (see Figure 5.2). Let $T = t_1 t_2 \ldots t_k$ be a substring of length k. In each input string S^j, H-ALIGN finds the best match to T in terms of Hamming distance, i.e., the k-long substring T^j of S^j that has the smallest number of mismatched characters with T. The substrings $T^1, \ldots, T^n$ (one of which is T itself) form the alignment A_T. After considering all $O(L)$ different substrings in the input, H-ALIGN obtains $O(L)$ alignments. The N_a alignments with the lowest entropy score are passed to the second phase. H-ALIGN runs in time $O(kL^2)$. Fortunately, a few simple heuristics, which we describe below, reduce the running time considerably with marginal impact on the quality of the results.

```
H-Align (I):
Foreach $k$-long substring $T$ of $S^1, \ldots, S^n$ do:
    $A_T \leftarrow \emptyset$.
    Foreach string $S^j$, $j = 1, \ldots, n$ do:
        Add to $A_T$ the best match in $S^j$ to $T$.
    $D_{A_T} \leftarrow$ Column distribution matrix of $A_T$.
    $H_{A_T} \leftarrow$ Entropy score of $D_{A_T}$.
Output $N_a$ alignments with lowest entropy score.
```

Figure 5.2: The basic alignment phase in HYDEN.

Let $A_h \subset A$ be an arbitrary subset of an alignment A, $|A_h| = N_h$. Provided that N_h is not too small, we can use A_h in order to estimate how well-conserved A is, or, in other words, we may assume that $H_{A_h} \approx H_A$. Thus, a more efficient version of H-ALIGN iterates all k-long substrings, and aligns only N_h input strings to each one. Then, the $N_{a'}$ substrings, whose alignments received the lowest (partial) entropy scores, are re-aligned against all n input strings, their full entropy score, H_A, is computed, and the best N_a ($\leq N_{a'}$) alignments are passed to the next stage. If all input strings have approximately the same length, then this efficient version of H-ALIGN runs in time $O(kL(\frac{N_h}{n}L + N_{a'}))$. Another improvement we applied exploits the fact that alignments obtained from highly overlapping substrings are very similar. Therefore, if the alignment we get from a substring $s_i \ldots s_{i+k-1}$ has a high entropy score, there is no point in checking the next substring: $s_{i+1} \ldots s_{i+k}$, as it is highly unlikely to yield good results, too. In fact, if the entropy score is very poor, we may decide to skip more than one substring. In practice, this simple idea reduced the running time of H-ALIGN by another factor of 2–4.

The second phase constructs two primers from each of the N_a alignments. Given an alignment A with a column distribution matrix D_A, HYDEN runs two heuristics — H-CONTRACTION and H-EXPANSION. These algorithms are generalizations of the CONTRACTION and EXPANSION approximation algorithms, respectively, to non-binary alphabets. H-CONTRACTION starts with a fully degenerate primer, and discards characters at degenerate positions with the smallest count in D_A until the primer reaches the required degeneracy, as shown in Figure 5.3. H-EXPANSION employs an opposite approach. It uses the substring $T \in A$, from which A was constructed, as an initial non-degenerate primer, and repeatedly adds to it a character with the largest count as long as its degeneracy does not exceed the threshold d, as detailed in Figure 5.4. Notice that the original EXPANSION algorithm repeats this procedure for each substring in A. However, early experiments demonstrated that if many of the input strings can be covered by a single primer, there is very little difference between primers obtained by expanding different substrings in A (data not shown). Therefore, in H-EXPANSION we chose to expand only one substring from each alignment. Finally, the second phase of HYDEN computes the coverage of the $2N_a$ primers it constructed, and selects the N_g ($\leq 2N_a$) primers that match the largest number of input strings (with up to e mismatches). The running time of the second phase of HYDEN is $O(N_a kL)$.

```
H-Contraction ($I; A$):
Sort the counts: $D_A(b_1, i_1) \leq D_A(b_2, i_2) \leq \ldots \leq D_A(b_{4k}, i_{4k})$.
$P \leftarrow$ Fully degenerate primer ; $j \leftarrow 1$.
**While** $d(P) > d$ and $j \leq 4k$ **do**:
    $P' \leftarrow P$ without character $b_j$ at position $i_j$.
    **If** $d(P') \neq 0$ **then** $P \leftarrow P'$.
    $j \leftarrow j + 1$.
Output $P$.
```

Figure 5.3: The H-CONTRACTION algorithm used by HYDEN.

```
*H-Expansion* $(I; A)$:
Sort the counts: $D_A(b_1, i_1) \geq D_A(b_2, i_2) \geq \ldots \geq D_A(b_{4k}, i_{4k})$.
Let $T$ be the substring from which $A$ was constructed.
$P \leftarrow T$ ; $j \leftarrow 1$.
**While** $j \leq 4k$ **do**:
    $P' \leftarrow P$ with character $b_j$ added at position $i_j$.
    **If** $d(P') \leq d$ **then** $P \leftarrow P'$.
    $j \leftarrow j + 1$.
Output $P$.
```

Figure 5.4: The H-EXPANSION algorithm used by HYDEN.

The final phase of HYDEN tries to improve the N_g primers found in the previous phase using a simple hill-climbing procedure, called H-GREEDY. Given a primer P, H-GREEDY checks whether it can remove a character in a degenerate position in P and add a different character in any position instead, so that the coverage of the primer increases. This process is repeated as long as coverage is improving (see Figure 5.5). Denote by r the number of iterations performed until a local maximum is reached. Then, the running time of H-GREEDY is $O(rk^3L)$. In our experiments, r was almost always below 5. In order to limit the running time in the general case, one could fix an upper bound $\bar{r}$ on the number of improvement iterations the algorithm performs, thereby setting the total running time of the third phase of HYDEN to $O(N_g\bar{r}k^3L)$.

```
*H-Greedy* $(I; P)$:
$P^* \leftarrow P$, *improved* $\leftarrow$ "yes".
**While** *improved* $=$ "yes" **do**:
    *improved* $\leftarrow$ "no".
    **Foreach** degenerate character $(b, i)$ in $P$ **do**:
        $P' \leftarrow P$ without character $b$ at position $i$.
        **Foreach** degeneracy $(b', i')$ not in $P$ **do**:
            $P'' \leftarrow P'$ with character $b'$ added at position $i'$.
            $m(P'') \leftarrow$ Coverage of $P''$.
            **If** $d(P'') \leq d$ and $m(P'') > m(P^*)$ **then** $P^* \leftarrow P''$.
    **If** $m(P^*) > m(P)$ **then** $P \leftarrow P^*$, *improved* $\leftarrow$ "yes".
Output $P$.
```

Figure 5.5: The greedy hill-climbing procedure used by HYDEN. $m(P)$ denotes the coverage of primer P.

HYDEN runs in total time of $O(kL(\frac{N_h}{n}L + N_{a'} + N_g \bar{r} k^2))$. Notice that the input parameters d and e are missing from the formula — the reason is that the performance depends linearly on $\log d$ and e, both of which are accounted for in the $O(k)$ factor. As we shall demonstrate in the next chapter, HYDEN is sufficiently fast for designing a primer of length $k \leq 30$ for a set of hundreds of DNA sequences, each 1Kbp long. Moreover, by modifying the various parameters, one can control the tradeoff between the running time of the program and the quality of the solution it provides. We report concrete running times and parameters in the next chapter.

HYDEN is a generalization of the $(k - \delta)$-approximation of MC-DPD* that we presented in Section 4.2. If a set of binary strings of length k is supplied to the program, and $e = 0$, the alignment phase does nothing (the strings are already aligned), the second phase yields the approximation (H-CONTRACTION is identical to CONTRACTION when $|\Sigma| = 2$), and the final greedy phase may further improve the solution. We have no theoretical guarantee on the performance of HYDEN in the general case, and, specifically, for genomic sequences of arbitrary length. Nevertheless, as we shall see, the results it produced in practice for the OR subgenome were highly satisfactory.

Chapter 6

Application: Deciphering the Human OR Subgenome

In this chapter, we describe an experiment for revealing the genomic sequences of many human olfactory receptor (OR, in short) genes. We shall focus on the role of HYDEN in this experiment, and analyze its performance. We first acquaint ourselves with a few basic facts on the OR subgenome, then describe the experimental setup and the results, and finally provide various analyses of the performance of HYDEN.

6.1 The OR Subgenome

The human sense of smell can detect millions of different odorants — volatile ligands with diverse chemical structures. Olfaction occurs when odorants bind with receptors of olfactory sensory neurons in the nose. The genes encoding odorant receptors form the largest super-family in the vertebrate genome, encompassing some 2%–3% of all human genes (see, e.g., [25]) — roughly 900 OR sequences were found in the first draft of the human genome, of which more than 50% are considered pseudogenes [14, 42]. Olfactory receptor genes are found in most human chromosomes [10, 36], and many of them are organized in large clusters [14, 42]. OR genes have a single coding exon of about 1Kbp, and code for seven-transmembrane domain proteins [4]. They have several highly conserved regions, primarily in transmembrane (TM) segments 2 and 7. By contrast, TM segments 4 and 5 show a high degree of variability — a crucial feature for recognizing a huge variety of odorants [33].

6.2 The DEFOG Experiment

HYDEN was studied and implemented as part of DEFOG — an experimental scheme for DEciphering Families Of Genes. DEFOG provides a powerful means for analyzing the composition of a large family of genes with conserved regions, and is thus especially useful in species for which little genomic data is available. In addition, DEFOG can be applied to analyze cDNA libraries of gene families. Given a subset of known gene sequences, HYDEN is used to design degenerate primer pairs. The primers are then used in PCR procedures to amplify fragments of genes, known as well as unknown, of the same family. The fragments are cloned, and an oligofingerprinting (OFP) process [6, 17, 29, 34] characterizes the clones by their patterns of hybridization with a series of very short (8-mer) oligonucleotides. The hybridization pattern of a clone is called its *fingerprint*. Another novel algorithm, called CLICK [37], clusters the clones into groups corresponding to the same gene according to their fingerprints. Finally, representatives from each cluster are sequenced and compared to the existing OR database. The DEFOG scheme is illustrated in Figure 6.1.

The newly revealed sequences can be used to design primer pairs for another cycle of DEFOG. Note, however, that only the region *between* the 5' and 3' primers is revealed by DEFOG — the part of a gene upstream (downstream) the 5' (3') primer is not amplified by the PCR, and the subsequences matched by the primers themselves are only partially informative, since all mismatch information is lost during PCR (the original gene subsequence is not amplified, but rather the primer sequence that matched it). Hence, sequences revealed by DEFOG convey new information only downstream (upstream) the 5' (3') primer, so primers for the next cycle of DEFOG are designed based on this region. Thus, with each round of DEFOG, the length of the amplified gene sequences decreases.

We applied the DEFOG scheme on the human OR subgenome. In a single experiment, it almost tripled the size of our initial OR repertoire, from 127 genes (the number of known OR genes when the experiment began) to 358. The extremely degenerate primers we designed proved very effective: They achieved high sensitivity, amplifying 300 unique OR genes, and extremely high specificity, yielding only 0.4% (4 out of 924) non-OR products. The combination of the OFP process and the CLICK clustering software allowed a low-redundancy sequencing — cluster analysis partitioned the 13,580 clones we obtained into 239 clusters and 121 singletons (single clone clusters), from which we sequenced only 924 (6.8%) clones. The full experimental details and results are reported elsewhere [11]. The DEFOG project is joint work with the groups of H. Lehrach (MPI Berlin) and D. Lancet (Weizmann). The rest of this section describes the execution of HYDEN as part of this project.

Our experiment began with an initial collection of 127 OR genes, whose full DNA sequences of size 1Kbp were known at the time [10]. This collection comprised our *training set*, on which HYDEN designed the primers. As explained earlier, the first phase of HYDEN con-

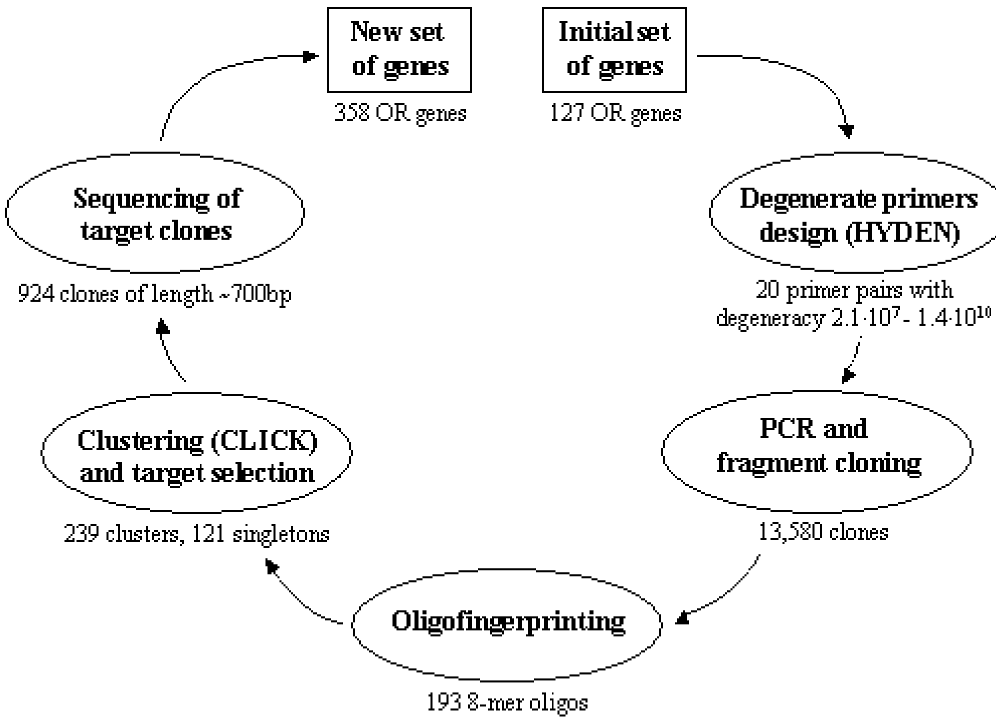
New set of genes
358 OR genes
Initial set of genes
127 OR genes
Degenerate primers design (HYDEN)
20 primer pairs with degeneracy 2.1·10^7 - 1.4·10^10
PCR and fragment cloning
13,580 clones
Oligofingerprinting
193 8-mer oligos
Clustering (CLICK) and target selection
239 clusters, 121 singletons
Sequencing of target clones
924 clones of length ~700bp

Figure 6.1: The DEFOG scheme. Starting with an initial set of known genes, DEFOG defines a series of computational and experimental steps that eventually allow low-redundancy sequencing of many new genes of the same family. The numbers beneath the boxes summarize the actual parameters in our DEFOG experiment on the human OR subgenome.

structs an alignment for each k-long substring in the input strings and computes its entropy score. Figure 6.2 shows the average entropy score for $k = 26$ at each position along the genes. The value at position p is an average of the entropy scores of the 127 alignments constructed for the substrings at position p in the genes. The local minima at positions 160 and 850 correspond to the conserved regions in TM2 and TM7, respectively. These positions make excellent sites for designing PCR primers that amplify a large portion of each gene — almost 700bp. The two other local minima, at positions 350 (TM3) and 700 (TM6), are more interior and a little less conserved.

In order to design both 5' and 3' primers, we ran HYDEN separately on the first and last 300bp of each OR gene. Altogether, we designed 13 primers, listed in Table 6.1 — 6 for the 5' side and 7 for the 3' side, of lengths $k = 26, 27$ and various degeneracies between $4,608$ and $442,368$. The primers on each side are quite similar to one another, and differ mainly in their degeneracy, except for four special primers — one pair (L9 and R110) was designed at different positions, closer to the 5' and 3' ends of the genes, and another pair (L20 and R20) was designed on a subset of genes that were poorly matched by the other primers.

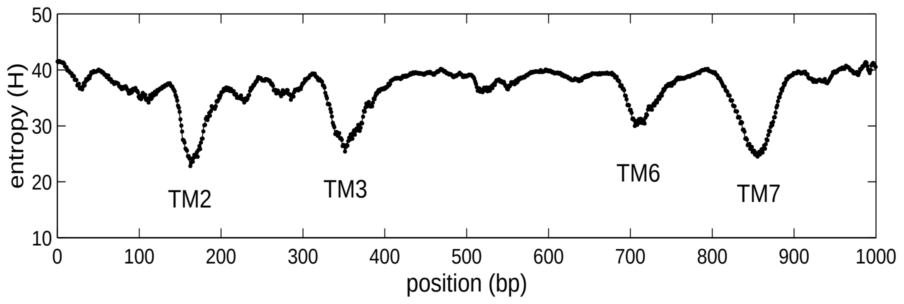

Figure 6.2: Average entropy score along the 127 OR genes in the training set. Transmembrane (TM) segments 2, 3, 6 and 7 contain highly conserved regions.

These four primers were constructed in order to "fish out" genes that, for some reason, are not amplified by the other primers. A typical run of HYDEN on 300bp segments of the 127 OR genes, with $k = 26$, $d = 20{,}000$, and $e = 2$ (and $N_h = 50$, $N_{a'} = 8{,}000$, $N_a = 3{,}000$, $N_g = 100$), takes approximately 10 minutes, distributed evenly among the three phases of the program, on a P4 1.4GHz PC with 256MB RDRAM. Except for the special primers, each primer matches $76\% - 90\%$ of the training-set genes with up to two mismatched bases.

From the 13 primers we designed, we selected 20 different pairs (see Table 6.2), and used them in PCR reactions. The degeneracy of a pair of primers (5' and 3') is defined as the product of the degeneracies of both primers. The degeneracy of the pairs we selected ranged between $2.1 \cdot 10^7$ and $1.4 \cdot 10^{10}$. To the best of our knowledge, this is the highest degeneracy ever used successfully in PCR reactions — extant applications usually use degeneracies lower than 10^5. We also experimented with even higher degeneracies (up to $2.2 \cdot 10^{11}$), but their yield was usually very poor, perhaps since the concentration of each individual primer is too low to allow successful PCR amplification. Most primer pairs covered $70\% - 80\%$ of the training-set genes with up to three mismatched bases in both sides combined (we used a threshold of three mismatches, since early experiments have shown that it predicts successful PCR amplification reasonably well — see Section 6.7).

6.3 The HORDE Test Set

After the publication of the first draft of the human genome, we analyzed the performance of the primers on all full-length OR sequences that were computationally detected in the draft. This set consisted of 719 genes [14] [1]. These genes served as a *test set*, with which we checked how well the coverage of our primers extends from the training set to a larger collection of genes. Note that 125 of the training-set genes are also in the test set, with slight changes.

[1]Sequences are available in the HORDE database at http://bioinformatics.weizmann.ac.il/HORDE

Side	Name	Primer sequence	Degeneracy	Coverage
5'	L5	`CTNCAHWCNCCHATGTAYTTYYTYCT`	4,608	107 (84%)
	L9 †	`ACNNTGVTNGGVAAYCTNCTCATYAT`	9,216	59 (46%)
	L10	`CTBCAYDNNCCHATGTAYTTYTTBCT`	10,368	112 (88%)
	L20 ‡	`CTYCANDVHCCCATGTAYYWYTTYBT`	20,736	110 (87%)
	L31	`CTBCAYDNNCCHATGTAYTTBTTBYT`	31,104	114 (90%)
	L131	`CTNCANWCNCCNATGTAYTTNYTNCTN`	131,072	110 (87%)
3'	R5	`TTYCTCARRSTRTADATNADNGGGTT`	4,608	97 (76%)
	R20 ‡	`TGKGABVHACANGTGBWRARRGCYTT`	20,736	79 (62%)
	R28	`TTBCKNARRSTRTADATVARRGGRTT`	27,648	105 (83%)
	R73	`TTBCKNARRSTRTADATNANRGGRTT`	73,728	109 (86%)
	R110 †	`YNCAGDRCHCYYTTNAYDTCYYTRTT`	110,592	57 (45%)
	R147	`RTTBCKNARNSTRTADATNARNGGGTT`	147,456	105 (83%)
	R442	`TTBCKNARRSTRTADATNANDGRRYT`	442,368	113 (89%)

Table 6.1: Degenerate primers designed by HYDEN on the training set of 127 OR genes. The primer sequences are specified using the NC-IUB nucleotide code (see Table 2.1). The last column shows the number (percentage) of genes (out of 127) that each primer matches with up to two mismatched nucleotides. † L9 and R110 were designed at different positions (TM1 and the end of TM7) than the other primers (TM2 and TM7). ‡ L20 and R20 were designed on a subset of genes that were poorly matched by all the other primers.

Figure 6.3 shows the 3-mismatches coverage of several primer pairs, both for the training set and the test set (Table 6.2 contains the full data). We excluded pairs that contain a special primer, in order to allow a fair comparison between pairs with different degeneracies. For the same reason, we included only pairs, in which the 5' and the 3' primers are of length 26 and have comparable degeneracy (to ensure that in all the pairs we compare the degeneracy is divided similarly between the two primers). The pairs that match these criteria are L5/R5, L10/R5, L5/R28, L10/R28, L31/R73, and L31/R442. The figure also shows the coverage of additional primers that were designed by HYDEN but were not used in the experiment.

As expected, primers with higher degeneracy have a larger coverage in both sets. Also apparent is the sharp and steady increase in the test-set coverage as the degeneracy increases — from 10% coverage for non-degenerate primers to 50%–65% for the primers we used and 74% for a pair with degeneracy $4 \cdot 10^{12}$. In practice, one cannot use arbitrarily high degeneracies, for two reasons. First, highly degenerate primers have low specificity, and so they might amplify many non-related sequences. This did not prove to be a problem even with the high degeneracies that we used — only 0.4% (4 out of 924) of the clones we sequenced were not OR genes. Second, as mentioned earlier, PCR gives a poor yield when the degeneracy is very high, which is what limited us to use primer pairs with degeneracy not higher than $1.4 \cdot 10^{10}$. Another conclusion from the above analysis is that the basic premise behind the DEFOG scheme proved valid: The training set was indeed a good representative set of the full set, in terms of primer properties, and facilitated the design of primers that matched hundreds of additional unknown genes.

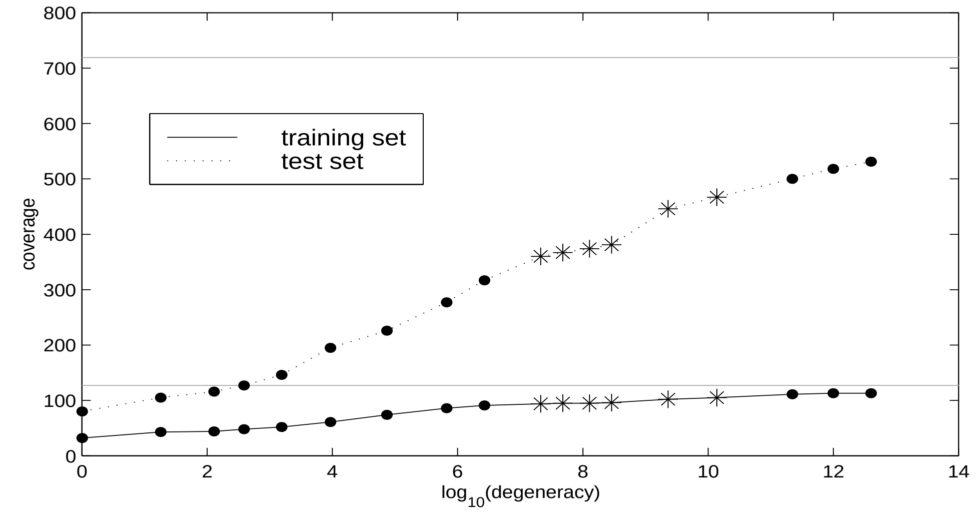

Figure 6.3: Training-set and test-set 3-mismatches coverage of primer pairs with various degeneracies. Primers that were actually used in the DEFOG experiment are marked by asterisks. The horizontal lines mark the size of the training and test sets.

6.4 Sequencing Efficacy

Table 6.2 summarizes the performance of the 20 primer pairs we used in the DEFOG experiment. Most of the primer pairs yielded a satisfactory number of clones (several hundreds). Exceptions are L131/R28 (181 clones) and L31/R442 (131 clones). The latter was the most degenerate primer pair for which we could obtain a reasonable yield. Since only 6.8% of the clones were sequenced, we do not know the full number of distinct genes each primer pair amplified. Thus, in order to evaluate how well the primers performed in practice, we computed their *sequencing efficacy* — the percentage of distinct genes that were obtained by each primer pair, out of the total number of clones sequenced for that pair (the seventh column in Table 6.2 divided by the sixth column). For 10 out of 12 primer pairs with degeneracy over 10^9, sequencing efficacy was $79\% - 93\%$, whereas for all 8 primers with lower degeneracy, it was $57\% - 79\%$.

Figure 6.4 shows the sequencing efficacy of several of the primer pairs we used, as a function of the degeneracy. As in the previous section, in order to allow a fair comparison between the primers, we depict only the pairs L5/R5, L10/R5, L5/R28, L10/R28, L31/R73, and L31/R442. Also shown in the figure is the number of *new* genes (with respect to the training-set) sequenced from each primer pair, as a percentage of the total number of clones sequenced for that pair. The correlation between this number and the sequencing efficacy is very apparent — for most primers, 70% – 90% of the genes we sequenced were new; for the six pairs shown in Figure 6.4, the ratio is much less variant — 72% – 75% of the genes were new.

Note that the sequencing efficacy, according to the way we compute it, depends not only on the performance of the primers, in terms of the number of genes they amplified, but also on the clustering and target selection procedures. For example, if CLICK assigned the clones of a certain gene to two or more clusters, instead of just one, then we may have sequenced multiple copies of that gene and the sequencing efficacy would have dropped. Furthermore, the 924 sequenced clones include 140 clones from six clusters, which we sequenced exhaustively in order to obtain statistics on the quality of the clustering analysis. The measured sequencing efficacy we report here is therefore lower than the true efficacy of the primers.

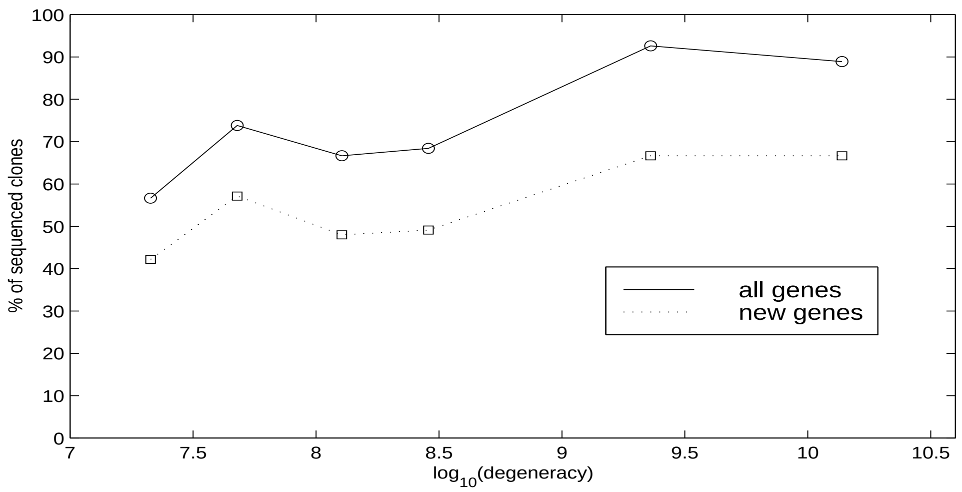

Figure 6.4: Sequencing efficacy of several primer pairs in the DEFOG experiment. The dotted line shows the number of *new* genes, i.e., genes that were not in the training set, as a percent of the number of sequenced clones.

6.5 Training-Set Size

An important question regarding the applicability of DEFOG in general, and HYDEN specifically, is their sensitivity to the size of the initial training set. In many projects, a large training set might not be available. Will HYDEN design good primers in such cases? Obviously, the answer depends on the characteristics of the set of genes, e.g., whether they contain

Primer	Degeneracy	3-mismatches coverage		Number of clones		Number of genes	
pair	($\times 10^6$)	training-set	test-set	total	sequenced	total	new
L5/R5*	21	73 %	50 %	1,730	173	98	73
L10/R5*	48	74 %	51 %	838	42	31	24
L5/R28*	127	74 %	52 %	901	75	50	36
L9/R20	191	31 %	13 %	431	43	25	14
L10/R28*	287	74 %	53 %	740	57	39	28
L5/R73	340	77 %	60 %	566	34	27	17
L5/R110	510	51 %	30 %	598	31	22	19
L31/R20	645	66 %	47 %	352	65	45	40
L9/R110	1,019	29 %	11 %	621	19	15	11
L9/R147	1,359	48 %	21 %	973	42	34	20
L10/R147	1,529	77 %	55 %	660	53	42	34
L5/R442	2,038	79 %	63 %	649	46	38	32
L31/R73*	2,293	80 %	62 %	1,033	27	25	18
L20/R147	3,058	77 %	51 %	747	67	43	34
L31/R110	3,440	55 %	31 %	426	25	21	19
L131/R28	3,624	76 %	57 %	181	14	12	11
L9/R442	4,077	54 %	26 %	748	28	20	14
L31/R147	4,586	78 %	56 %	564	28	26	18
L10/R442	4,586	80 %	63 %	691	46	37	26
L31/R442*	13,759	82 %	65 %	131	9	8	6
Total	—	93 %	76 %	13,580	924	300	231

Table 6.2: Primer pairs used in the DEFOG experiment on the human OR subgenome. The second column specifies the combined degeneracy of the two primers, in millions. The third and fourth columns are the percentage of genes, out of the training set (127 genes) and the test set (719 genes) respectively, that match the primer pair with up to 3 mismatched bases. The fifth column specifies the number of clones we obtained from the amplified PCR fragments, and the sixth column is the number of representative clones that were selected and successfully sequenced. The last two columns are the number of distinct genes each primer pair yielded — total number of genes, and new genes (that are not in the training set). * Pairs in which both primers were of length 26 with roughly equal degeneracy, and neither one of them is a special primer. The performance of these primer pairs is compared in Figures 6.3 and 6.4.

conserved regions for 5' and 3' primers, and how the similarity in those regions is distributed between the different genes. It also depends on the degree by which the training set represents the whole set, i.e., whether we are lucky enough to have representatives of many types of genes in the set.

In order to determine whether DEFOG would have yielded satisfactory results on the human OR subgenome if the initial set of known genes was substantially smaller, we tested how well HYDEN performs for different sizes of training sets. Given a size s, we selected a random training set of s genes from the test set of 719 currently known OR genes. We then used HYDEN on this subset to design primers of length 26, and computed their test-set coverage, i.e., the number of test-set genes that are matched by the primers with up to 3 mismatches in both sides combined. For each s, we repeated this test 20 times, and computed the average coverage and its standard deviation. Figure 6.5 shows the results for various values of s between 20 and 719, and two degeneracies: $2.1 \cdot 10^7$ ($4,608$ in both sides) and $2 \cdot 10^9$ ($27,648$ in the 5' side, $73,728$ in the 3' side). For both degeneracies, there is a sharp increase in the coverage as s grows from 20 to 60, and a very small improvement for values of s larger than 100. Even for $s = 30$ (merely 4% of the test set), HYDEN provides good primers with degeneracy $2 \cdot 10^9$, covering 54% of the test set, on average. In comparison, when HYDEN is given all 719 genes as its training set, it designs primers with the above degeneracy that cover 64% of the genes.

In our DEFOG experiment, the training set was fairly large ($s = 127$), and we would probably not have achieved significantly better results had it been larger. Not surprisingly, the standard deviation of the coverage diminishes as s increases. For small values of s, there is a high probability that the training set contains representatives of only some of the types of OR genes, preventing HYDEN from designing primers that match other types. Still, the standard deviation is quite small, and similar for both degeneracies we checked: $22 - 30$ ($3\% - 4\%$ of 719) for $30 \leq s \leq 60$, and $13 - 14$ (1.9% of 719) for $s = 100$. Thus, in the case of the human OR genes, DEFOG would have probably produced satisfactory results even if the training set consisted of only 30 genes; for a training set of 60 genes, the results would have been similar to those we obtained. In general, we believe that DEFOG can be successfully applied to any gene family, for which a small training set exists, provided that the genes exhibit high conservation in sites that are appropriate for 5' and 3' primers.

A caveat that applies both to the analysis above regarding the OR genes and to other families is that it is based on the assumption that the training set is obtained by random sampling from the family. In practice, that set is not completely random. The training set of OR genes, for example, was obtained by a combination of detection of ORs in papers and in large-scale sequencing databases (which yield a roughly random sample), and small-scale collection of ORs based on low-degeneracy primers (which is highly non-random).

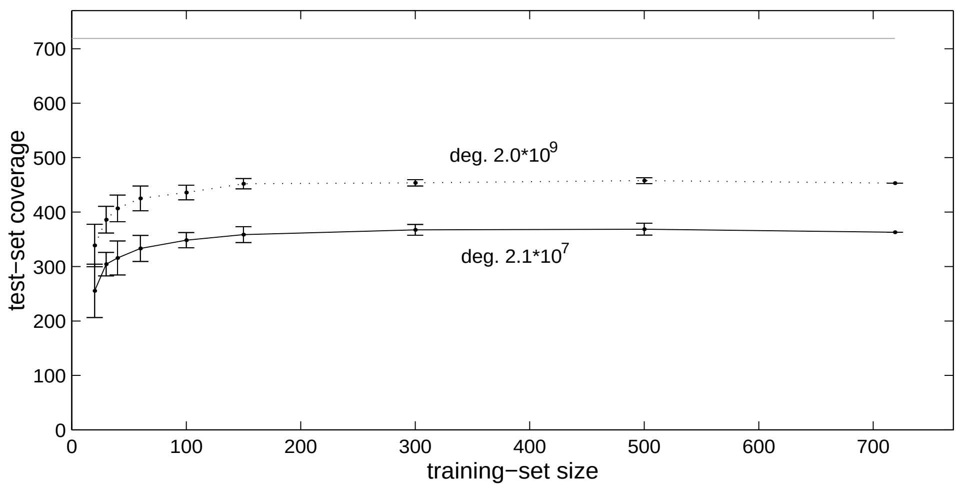

Figure 6.5: Test-set coverage as a function of the size of the training set. Random subsets of various sizes were selected from the 719 test-set genes. HYDEN was run on each subset to design primer pairs of degeneracies $2.1 \cdot 10^7$ and $2 \cdot 10^9$. Each experiment was repeated 20 times. The graphs show the average 3-mismatches test-set coverage of the primers, and the standard deviation (vertical bars indicate ±1 std.). The horizontal line marks the size of the test set.

6.6 Primers for the OR Subgenome

We ran HYDEN on the 719 fully-known OR genes that were extracted from the human genome draft, and assembled a new list of primer pairs with various degeneracies. The primers are listed in Table 6.3. In accordance with the results from the previous section, the coverage of these primers is larger by only 2% than that of primers with similar degeneracy designed on the original training set of 127 genes — indeed, the new primers are very similar to those we used in the OR experiment (compare Tables 6.1 and 6.3). We recommend that the new primers be used in future PCR procedures on olfactory genes in the human genome or in related species.

5' primer	3' primer	Degeneracy	Coverage
`CTNCAYDMNCCCATGTAYTTYYTBCT`	`TTYCTCARDSTRTAGATNANDGGRTT`	$2.1 \cdot 10^7$	52 %
`YTNCAYHMNCCHATGTAYTWYTTBCT`	`KTYCTNAVRSTRTARATNANDGGRTT`	$2.0 \cdot 10^9$	64 %
`CTBCAYNVNCCHATGTAYTWYYTYCT`	`TTYYKNAVRSTRTADAYNANDGGRTT`	$1.2 \cdot 10^{10}$	67 %

Table 6.3: Degenerate primer pairs for the human OR subgenome. Primers of length 26 were designed by HYDEN on the test set of 719 OR genes. Sequences are specified using the NC-IUB nucleotide code. The third column shows the combined degeneracy, and the last column is the percent of test-set genes that are covered with up to 3 mismatches.

In some experiments, a full, or nearly-full coverage of the family of genes is particularly important. In such cases, several degenerate primer pairs need to be used (recall the MP-DPD problem). To assist in the design of a small set of primers that together cover a very large portion of a given training set, we added an external loop to HYDEN. After the first pair of primers is computed, all the input sequences that are covered by the primers are discarded, and the program uses the remaining sequences as the training set for designing another pair of primers, and so on. Applying this procedure to the 719 human OR genes, we designed 10 primer pairs with combined degeneracy $\sim 10^9$, shown in Table 6.4. We observed a very sharp decrease in the coverage of the primers — while the first pair covers 62% of the whole set, the second pair covers only 33% of the remaining 276 genes, and the third pair covers merely 13% of the remaining 184 genes. The accumulated coverage of all 10 primer pairs is 88%, somewhat disappointing given the coverage of the first pair. In other gene families, the situation might be different, and fewer primers may suffice for almost full coverage.

	5' primer	3' primer	Coverage	
1	`CTBCAYDVNCCHATGTAYTTYYTYCT`	`TTYCKNARRSTRTANATNANDGGRTT`	443	62 %
2	`CCCATGTAYTWYTTBCTBDSYAWBYT`	`TGDGARVYRCADGTRBHRARDGCYTT`	92	74 %
3	`YTNCACHCVCCHATGTAYTTYYTBYT`	`TCYTTRTTYCWNARRSTDTARAYNAN`	23	77 %
4	`GSCTGYMTBRYHCAGMTSTWYTTCWT`	`SYRTARAYNAKRGGRTTNADCANWGG`	22	81 %
5	`ATGTACTWYTTCCTBNSYHWYYTSTC`	`TGRGAGVYRCARGTRVHVADVRCYTT`	14	82 %
6	`YTBCRCNMVCCCATGTWYTWYTTYCT`	`TTGGTBYKVABVCYRTWRAYRAWRGG`	10	84 %
7	`TTGGTBYKVABVCYRTWRAYRAWRGG`	`RKMRKNKARRATGAVHMCATAGGARR`	7	85 %
8	`CYCRYYTBCAYWVMCCYATGTAYTTY`	`YSTCYBYRKTYYTCADRCTGTADAYV`	6	86 %
9	`CTBCAYAYWCCHATGTAYYWHTTBYT`	`RTGGGMGVHRCAVGTVGARWARGYYT`	9	87 %
10	`TGTACHWCYTCCTBDGVMWYYTSTCC`	`CACAGCTVBCAGRAYVANRDTRWMRR`	7	88 %

Table 6.4: Degenerate primer pairs of length 26 and combined degeneracy 10^9 designed by HYDEN to cover together the test set of 719 OR genes. The fourth column shows the number of genes that are covered by each primer pair (with up to 3 mismatches) and not covered by the previous primers. The last column is the cumulative percent of covered test-set genes.

6.7 Short DEFOG

The DEFOG scheme comprises four laboratory procedures — PCR, cloning, oligofingerprinting (OFP) and sequencing. The fingerprints obtained in the third stage are fed to the clustering program, CLICK, and representatives of each cluster are selected for sequencing. As we have shown, this scheme results in a low-redundancy sequencing — in our DEFOG experiment, we sequenced 924 clones and revealed 300 (32.5%) distinct OR genes. The sequencing efficacy of many of the primer pairs was 80% or more.

An interesting question is to what extent do the OFP and clustering processes contribute to the low sequencing redundancy. Perhaps in some cases, when the primers amplify many genes evenly, we could apply a simpler scheme, without the OFP and clustering phases. In this simplified version of DEFOG, which we call *"Short-DEFOG"*, clones are selected for sequencing arbitrarily, rather than based on the fingerprints clustering. The advantage of this scheme is clear — it skips the OFP phase. On the down side, we might sequence many copies of the same gene (or a small set of genes) again and again, which will require us to sequence a very large number of clones in order to get a reasonable coverage of the target set of genes.

Prior to the full DEFOG experiment, we ran a small round of Short-DEFOG on the human OR subgenome together with the group of D. Lancet (Weizmann). As in the full experiment, we used the 20 primer pairs that were designed on the 127 training-set genes. A total of 227 fragments were successfully cloned and sequenced. This set contained 102 unique ORs, of which 70 were new (i.e., not in the training set). Table 6.5 contains detailed statistics for each primer pair separately. Analysis of the sequences of the known genes and the primers that amplified them revealed that in roughly 50% of the cases there were no mismatches between the gene sequences and the primers, in 40% there was exactly one mismatch, and only in 3% there were more than three mismatches (in both sides combined).

The overall sequencing efficacy of the Short-DEFOG round was 45% (102 distinct genes out of 227 clones), much higher than the efficacy of the full DEFOG experiment, in which it was 32.5% (300 out of 924). Obviously, the comparison is not fair, since the sequencing efficacy decreases with the number of clones sequenced. Moreover, as mentioned in Section 6.4, some of the clones we sequenced in the DEFOG experiment were selected merely to obtain data for clustering analysis, so the real efficacy is higher than 32.5%. Nevertheless, the results of the Short-DEFOG round indicate that the primers HYDEN designed were not biased towards any specific set of genes, allowing for low-redundancy sequencing even without the OFP and clustering procedures. Figure 6.6 shows a histogram of the number of copies we sequenced of each gene. Although targets for sequencing were chosen arbitrarily, 92% of the genes were sequenced at most four times. It is therefore not surprising that in the full DEFOG experiment, CLICK partitioned the $13,580$ fingerprints into many small clusters, namely, 360 clusters (including singletons), of which about $\frac{2}{3}$ had a size of at most 20, and 86% contained no more than 100 members.

We recommend applying Short-DEFOG when partial coverage of the target set of genes suffices. If nearly-full coverage is required, or if the primers are biased towards a small subset of genes, then the complete DEFOG cycle should be carried out.

Primer pair	Degeneracy ($\times 10^6$)	3-mismatches coverage training-set	test-set	Number of clones sequenced	Number of genes total	new
L5/R5	21	73 %	50 %	36	30	23
L10/R5	48	74 %	51 %	15	13	8
L5/R28	127	74 %	52 %	10	8	6
L9/R20	191	31 %	13 %	2	2	1
L10/R28	287	74 %	53 %	29	22	15
L5/R73	340	77 %	60 %	6	5	2
L5/R110	510	51 %	30 %	7	6	4
L31/R20	645	66 %	47 %	6	5	5
L9/R110	1,019	29 %	11 %	12	8	4
L9/R147	1,359	48 %	21 %	12	10	6
L10/R147	1,529	77 %	55 %	9	5	3
L5/R442	2,038	79 %	63 %	11	10	7
L31/R73	2,293	80 %	62 %	11	11	8
L20/R147	3,058	77 %	51 %	6	6	5
L31/R110	3,440	55 %	31 %	8	5	4
L131/R28	3,624	76 %	57 %	3	2	2
L9/R442	4,077	54 %	26 %	14	12	6
L31/R147	4,586	78 %	56 %	15	11	4
L10/R442	4,586	80 %	63 %	9	6	4
L31/R442	13,759	82 %	65 %	6	6	6
Total	—	93 %	76 %	227	102	70

Table 6.5: Primer pairs used in the Short-DEFOG experiment on the human OR subgenome. The second, third and fourth columns are identical to Table 6.2. The fifth column specifies the number of fragments that were successfully cloned and sequenced. The last two columns are the number of distinct genes each primer pair yielded — total number of genes, and new genes.

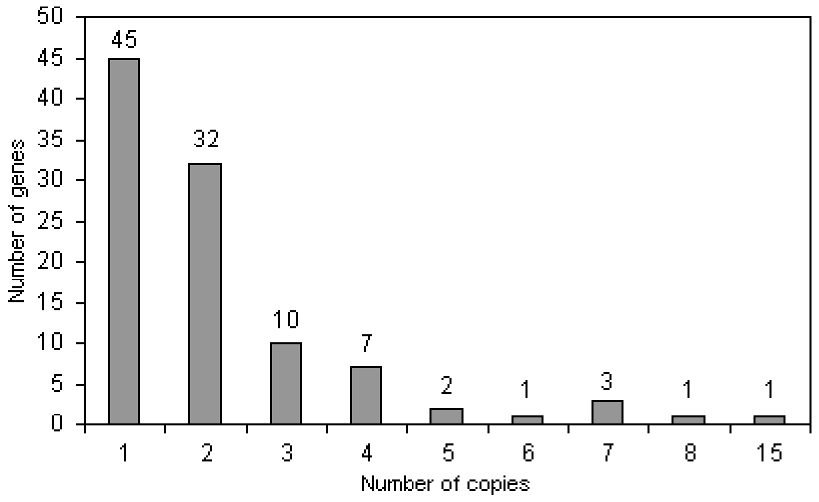

Figure 6.6: Histogram of the number of copies sequenced per gene in the Short-DEFOG experiment. Although target clones for sequencing were chosen arbitrarily (and not by their fingerprints, as in the full DEFOG scheme), most genes were sequenced very few times, indicating that the primers were unbiased.

6.8 The Canine Olfactory Subgenome

Encouraged by the results we obtained for the human OR subgenome, we recently launched a project with the group of D. Lancet (Weizmann) for analyzing the canine OR subgenome. We use two approaches: data mining in the Celera $\times 1$ sequence coverage of the dog genome, and Short-DEFOG. Since very few canine OR genes were fully known at the time, we ran HYDEN on the set of 719 *human* ORs, and designed several primer pairs with degeneracy between $6.4 \cdot 10^7$ and $2.2 \cdot 10^{10}$. Despite the significant differences between the human and canine olfactory systems, the human-based primers amplified many ORs from the dog genome. Preliminary results indicate that the 1200 clones we sequenced contain 246 (20.5%) unique canine OR genes (the full dog OR repertoire is estimated to contain some 1300 genes). About 15% of these genes are pseudogenes, similar to the ratio in mouse (20%) [40, 41], but far from the ratio in human ($> 50\%$) [14, 42]. This reflects the fact that both dog and mouse are macrosmatic animals, i.e., have a very acute sense of smell, whereas human is microsmatic. The full details of our work on the canine olfactory subgenome appear in [12]. We believe that this project demonstrates that DEFOG can be applied to study an unsequenced genome using degenerate primers designed according to a related species.

Chapter 7

Summary

In this thesis, we introduced DPD — a combinatorial optimization problem aiming for optimal design of degenerate primers. We defined several variants of the problem, and studied their computational complexity. We developed approximation algorithms for MC-DPD, a simplified version of DPD, with binary input strings, and implemented HYDEN, an efficient heuristic for the general case. We executed HYDEN as part of an experiment for sequencing the human olfactory subgenome. HYDEN proved quite effective in designing highly degenerate and yet highly specific primers.

7.1 Future Work

On the theoretical side, one may wish to design better approximation algorithms for MC-DPD in order to obtain a better approximation ratio and/or faster running time. Tighter inapproximability bounds could close the gap from the other, less desirable, direction. Another important advance would be to generalize the algorithms to cope with arbitrary length input strings over non-binary alphabets and allow mismatches between the primer and the strings. Approximation algorithms for other DPD variants we defined, namely MD-DPD and MP-DPD, could also have practical contribution.

On the practical side, a more realistic primer-gene matching model, which takes into account biological aspects of the PCR procedure, could yield primers with greater sensitivity. It is known that mismatches at the 3' terminus are more detrimental to PCR than internal mismatches [23], and that different types of mismatches have different effects on the reaction, e.g., A:C is less disruptive than A:G [24]. In addition, a situation where one primer is complementary to itself or to another primer should be avoided, since it leads to a competition among the primers and the sequences and greatly reduces the efficiency of the PCR. Other factors that should be considered are the GC content and melting temperature of the primers.

The first phase of HYDEN locates many conserved blocks in the given sequences. Instead, we could perform this step using some other available software for computing ungapped local multiple alignments, such as ClustalW [38] or BlockMaker [15]. For each block found in the first phase, HYDEN designs primers using heuristics based on the CONTRACTION and EXPANSION approximation algorithms. It would be interesting to implement the CONTRACTION-X algorithm (or, for practical applications, a generalization of it to non-binary alphabets) and compare its performance to that of CONTRACTION. Theoretically, at least, CONTRACTION-X should produce primers with larger coverage.

We hope to exploit the utility of degenerate primers on other gene families and other species. In addition to the aforementioned experiment for extracting OR genes from the dog genome, we are currently involved in a project that uses degenerate primers for identifying a subfamily of tyrosine kinases from cDNA libraries. HYDEN is also being employed by several other labs for various tasks.

Bibliography

[1] The HYDEN homepage: `http://www.math.tau.ac.il/∼rshamir/hyden/HYDEN.htm`.

[2] T.L. Bailey and C. Elkan. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21(1-2):51–80, 1995.

[3] M. Blum, R.W. Floyd, V. Pratt, R.L. Rivest, and R.E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7:448–461, 1973.

[4] L. Buck and R. Axel. A novel multigene family may encode odorant receptors: A molecular basis for odor recognition. *Cell*, 65:175–187, 1991.

[5] J. Buhler and M. Tompa. Finding motifs using random projections. *Journal of Computational Biology*, 9(2):225–242, 2002.

[6] M.D. Clark, G.D. Panopoulou, D.J. Cahill, K. Bussow, and H. Lehrach. Construction and analysis of arrayed cDNA libraries. *Methods in Enzymology*, 303:205–233, 1999.

[7] I. Dinur and S. Safra. The importance of being biased. In *Proc. 34th ACM Symp. on the Theory of Computing (STOC 2002)*, pages 33–42, Montreál, Canada, 2002.

[8] K. Doi and H. Imai. Greedy algorithms for finding a small set of primers satisfying cover length resolution conditions in PCR experiments. In *Proc. 8th Workshop on Genome Informatics*, pages 43–52, Tokyo, Japan, 1997.

[9] D. Dor and U. Zwick. Selecting the median. *SIAM Journal on Computing*, 28(5):1722–1758, 1999.

[10] T. Fuchs, G. Glusman, S. Horn-Saban, D. Lancet, and Y. Pilpel. The human olfactory subgenome: From sequence to structure and evolution. *Human Genetics*, 108:1–13, 2000.

[11] T. Fuchs, B. Malecova, C. Linhart, R. Sharan, M. Khen, R. Herwig, D. Shmulevich, R. Elkon, M. Steinfath, J.K. O'Brien, U. Radelof, H. Lehrach, D. Lancet, and R. Shamir. DEFOG: A practical scheme for deciphering families of genes. *Genomics*, 80(3):295–302, 2002.

[12] T. Fuchs, Z. Olender, M. Khen, C. Linhart, R. Shamir, F. Kalush, and D. Lancet. The canine olfactory subgenome. Manuscript, 2002.

[13] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[14] G. Glusman, I. Yanai, I. Rubin, and D. Lancet. The complete human olfactory subgenome. *Genome Research*, 11:685–702, 2001.

[15] S. Henikoff, J.G. Henikoff, W.J. Alford, and S. Pietrokovski. Automated construction and graphical presentation of protein blocks from unaligned sequences. *Gene*, 163:GC:17–26, 1995.

[16] G.Z. Hertz and G.D. Stormo. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15(7/8):563–577, 1999.

[17] R. Herwig, A.O. Schmidt, M. Steinfath, J. O'Brian, H. Seidel, S. Meier-Ewert, H. Lehrach, and U. Radelof. Information theoretical probe selection for hybridisation experiments. *Bioinformatics*, 16:890–898, 2000.

[18] J. Håstad. Some optimal inapproximability results. In *Proc. 29th ACM Symp. on the Theory of Computing (STOC 1997)*, pages 1–10, El Paso, Texas, 1997.

[19] J.D. Hughes, P.W. Estep, S. Tavazoie, and G.M. Church. Computational identification of cis-regulatory elements associated with groups of functionally related genes in saccharomyces cerevisiae. *J. Mol. Biol.*, 296(5):1205–1214, 2000.

[20] D.S. Johnson, 2002. Private communication.

[21] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New-York, 1972.

[22] U. Keich and P.A. Pevzner. Finding motifs in the twilight zone. In *Proc. 6th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2002)*, pages 195–204, 2002.

[23] 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 and Appl.*, 3:S39–47, 1994.

[24] 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 Research*, 18:999–1005, 1990.

[25] D. Lancet and N.R. Ben-Arie. Olfactory receptors. *Current Biol.*, 3:668–674, 1993.

[26] C.E. Lawrence, S.F. Altschul, M.S. Boguski, J.S. Liu, A.F. Neuwald, and J.C. Wootton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, 1993.

[27] C. Linhart and R. Shamir. The degenerate primer design problem. *Bioinformatics*, 18, Suppl. 1:S172–S180, 2002.

[28] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. In *Proc. 25th ACM Symp. on the Theory of Computing (STOC 1993)*, pages 286–293, San Diego, California, 1993.

[29] S. Meier-Ewert, J. Lange, H. Gerst, R. Herwig, A. Schmitt, J. Freund, T. Elge, R. Mott, B. Herrmann, and H. Lehrach. Comparative gene expression profiling by oligonucleotide fingerprinting. *Nucleic Acids Research*, 26:2216–2223, 1998.

[30] Nomenclature Committee of the International Union of Biochemistry (NC-IUB). Nomenclature for incompletely specified bases in nucleic acid sequences — recommendations 1984. *Biochemical Journal*, 229:281–286, 1985.

[31] W.R. Pearson, G. Robins, D.E. Wredgs, and T. Zhang. On the primer selection problem in polymerase chain reaction experiments. *Discrete Applied Mathematics*, 71:231–246, 1996.

[32] P.A. Pevzner and S. Sze. Combinatorial approaches to finding subtle signals in DNA sequences. In *Proc. 8th International Conference on Intelligent Systems for Molecular Biology (ISMB 2000)*, pages 269–278, 2000.

[33] Y. Pilpel and D. Lancet. The variable and conserved interfaces of modeled olfactory receptor proteins. *Protein Science*, 8:969–977, 1999.

[34] U. Radelof, S. Hennig, P. Seranski, M. Steinfath, J. Ramser, R. Reinhardt, A. Poustka, F. Francis, and H. Lehrach. Preselection of shotgun clones by oligonucleotide fingerprinting: An efficient and high throughput strategy to reduce redundancy in large-scale sequencing projects. *Nucleic Acids Research*, 26:5358–5364, 1998.

[35] T.M. Rose, E.R. Schultz, J.G. Henikoff, S. Pietrokovski, C.M. McCallum, and S. Henikoff. Consensus-degenerate hybrid oligonucleotide primers for amplification of distantly related sequences. *Nucleic Acids Research*, 26:1628–1635, 1998.

[36] S. Rouquier, S. Taviaux, B.J. Trask, V. Brand-Arpon, G. van den Engh, J. Demaille, and D. Giorgi. Distribution of olfactory receptor genes in the human genome. *Nature Genetics*, 18:243–250, 1998.

[37] R. Sharan and R. Shamir. CLICK: A clustering algorithm with applications to gene expression analysis. In *Proc. 8th International Conference on Intelligent Systems for Molecular Biology (ISMB 2000)*, pages 307–316, 2000.

[38] J.D. Thompson, D.G. Higgins, and T.J. Gibson. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

[39] O.V. Vishnevsky, O.A. Podkolodnaya, and V.N. Babenko. Search for degenerate oligonucleotide motifs in transcription factor binding sites and eukaryotic promoters (computer system ARGO). In *Proc. 1st International Conference on Bioinformatics of Genome Regulation and Structure*, pages 144–146, 1998.

[40] J.M. Young, C. Friedman, E.M. Williams, J.A. Ross, L. Tonnes-Priddy, and B.J. Trask. Different evolutionary processes shaped the mouse and human olfactory receptor gene families. *Human Molecular Genetics*, 11(5):535–546, 2002.

[41] X. Zhang and S. Firestein. The olfactory receptor gene superfamily of the mouse. *Nature Neuroscience*, 5(2):124–133, 2002.

[42] S. Zozulya, F. Echeverri, and T. Nguyen. The human olfactory receptor repertoire. *Genome Biology*, 2:RESEARCH0018, 2001.