The Degenerate Primer Design Problem:

Theory and Applications

Chaim Linhart* Ron Shamir

School of Computer Science,

Tel Aviv University,

Tel Aviv 69978, ISRAEL

E-mail: {chaiml,rshamir}@post.tau.ac.il

Voice: +972-3-640-5394 , Fax: +972-3-640-5384

March 2004

* Corresponding author

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 several applications, one of which is 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.

Keywords: Degenerate Primers for PCR, Complexity, NP-Hardness, Approximation Algorithms, Olfactory Receptor Genes.

1 Introduction

Polymerase chain reaction, or PCR (Mullis et al., 1986), is a ubiquitous technique which amplifies a specific region of DNA, so that enough copies of that region are available to be adequately tested or sequenced. In order to use PCR, one must 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.

A PCR primer sequence is called degenerate if some of its positions have several possible bases (Kwok et al., 1994). 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 work we study the problem of designing primers of high degeneracy.

Suppose 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 problem Degenerate Primer Design (DPD).

Our need to study DPD arose in a joint 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) (Pearson et al., 1996), 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., (Doi and Imai, 1997)). However, for large gene families, the number of primers needed to cover a sufficient 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 (Rose et al., 1998) and DePiCt (Wei et al., 2003) are programs for designing degenerate primers for multiply-aligned protein sequences. CODEHOP constructs a pair of primers for each given multiple alignment. 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. DePiCt clusters the sequences using a simple similarity score, and then designs a pair of primers for each cluster by translating conserved blocks of amino-acids into nucleotides. Another algorithm for designing multiple degenerate primer pairs, called MIPS (Souvenir et al., 2003), was developed very recently in the context of SNP genotyping. (Both DePiCt and MIPS were developed following our initial introduction of DPD in (Linhart and Shamir, 2002)). Souvenir et al. define two variants of the Multiple Degenerate Primer Design problem (MDPD), in which the goal is to find a minimum number of primers that together match all the input sequences. MIPS uses a beam-search technique to progressively construct a set of primers until all sequences are covered.

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 (Bailey and Elkan, 1995), Random Projections (Buhler and Tompa, 2002), CONSENSUS (Hertz and Stormo, 1999), AlignACE (Hughes et al., 2000), Multiprofiler (Keich and Pevzner, 2002), Gibbs Sampler (Lawrence et al., 1993), WINNOWER (Pevzner and Sze, 2000)) 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 (Vishnevsky et al., 1998)), 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.

In this work we study the DPD problem from theoretical and practical perspectives. We define and study 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 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 (Glusman et al., 2001). We also report results of two other projects that utilized HYDEN: an experiment for deciphering the canine olfactory subgenome, and a study on the degeneration of the olfactory repertoire in primates. HYDEN is freely available for academic use (http://www.math.tau.ac.il/∼rshamir/hyden/HYDEN.htm).

The remainder of the work is organized as follows. In Section 2 we give formal definitions of the problems. Section 3 gives hardness results and polynomial algorithms for several problem variants. In Section 4 we give approximation algorithms. Section 5 describes the HYDEN program, and Section 6 presents the actual performance of HYDEN in the OR project. A summary and directions for further research are given in Section 7. A preliminary version of this study appeared as an extended abstract in (Linhart and Shamir, 2002). The application of HYDEN to the OR subgenome was reported in (Fuchs et al., 2002).

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 predefined 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.

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, $\Sigma = \{\text{A,C,G,T}\}$. 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 $P^* = \{\text{A}\}\{\text{C,G}\}\{\text{A,C,G,T}\}\{\text{G}\}\{\text{T}\}$ 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, $P^* = \text{A}\{\text{C,G}\}*\text{GT}$. An alternative way to describe a primer is using the NC-IUB (Nomenclature Committee of the International Union of Biochemistry) nucleotide code (NC-IUB, 1985), also termed the IUPAC (International Union of Pure and Applied Chemistry) nucleotide code. 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: $\lceil \log_{|\Sigma|} d(P) \rceil \leq \delta(P) \leq \lfloor \log_2 d(P) \rfloor$.

A primer $P^1 = p_1^1 p_2^1 \ldots p_k^1$ is a *sub-primer* of a primer $P^2 = p_1^2 p_2^2 \ldots p_k^2$ of the same length, if $\forall i, 1 \leq i \leq k$, $p_i^1 \subseteq p_i^2$. 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 $p_i^{12} = p_i^1 \cup p_i^2$.

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 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 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., $m = n$.

Figure 1 here

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 (Sections 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 (Kwok et al., 1994). 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.

In MP-DPD we assume that all the input strings are of the same length as the primers. If we remove this constraint, i.e., allow the strings to have arbitrary length, we get a more general problem. This variant of DPD, called Multiple DPD (MDPD), is studied in (Souvenir et al., 2003).

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 Section 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 (Kwok et al., 1994). Our theoretical results focus on the simple, restricted DPD variants. As we will see in the next section, even those are hard. Our heuristics, though, address most of the realistic issues satisfactorily.

3 Complexity

In this section 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

DPD 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 Section 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. 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.*

■

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 ((Karp, 1972), (Garey and Johnson, 1979, 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 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. Figure 2 here

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 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.

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

The proof of the theorem is based on a reduction from *Minimum Set Cover* (MSC) ((Karp, 1972), (Garey and Johnson, 1979, SP5)). Using this reduction and a known hardness result of MSC, we can also show that it is difficult to approximate the number of degenerate positions in an optimal primer for MD-DPD:

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.*

The full proofs of Theorem 14 and Corollary 15 are given in (Linhart, 2002).

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.

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

To prove the theorem we use a reduction from *Minimum Vertex Cover* ((Karp, 1972), (Garey and Johnson, 1979, GT1)). Again, this allows us also to prove that it is difficult to approximate the number of degenerate positions in MD-EDPD.

Corollary 17 *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.*

The full proofs of Theorem 16 and Corollary 17 are given in (Linhart, 2002).

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 18 *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) ((Garey and Johnson, 1979, 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:

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

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 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 $p(l)$.

Figure 3 here

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 $S_{i_1}, \ldots, S_{i_u}$:

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

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 $\epsilon > 0$ (Garey and Johnson, 1979). 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 (Johnson, 2002). Therefore, we cannot apply the L-reduction to prove that MP-DPD is hard to approximate.

A generalized version of MP-DPD, in which the input strings may have arbitrary length, was shown to be $\mathcal{NP}$-Hard in (Souvenir et al., 2003). Our result is stronger: even if we limit all the strings to have the same length as the desired primers, the problem is $\mathcal{NP}$-Complete.

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 (Pearson et al., 1996), 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 (Doi and Imai, 1997).

4 Approximation Algorithms

In this section 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 Sections 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,...,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 '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 19 *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 shall 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 $S^j = s_1^j s_2^j \ldots s_k^j$ the j-th input string, $1 \le j \le n$, then:

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

Let $P^o = p_1^o p_2^o \ldots p_k^o$ 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, $\forall b \notin p_i^o$, $D(b,i) \leq u_o$, and for each non-degenerate position i in P^o, $D(p_i^o, i) \geq m_o$. 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 $v(i_1) \geq v(i_2) \geq \ldots \geq v(i_k)$ be the leading values in D, sorted from largest to smallest. The following lemma follows from the discussion above.

Lemma 20 *If P^o covers m_o strings, then $v(i_{k-\delta}) \geq m_o$.*

■

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.:

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

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 illustrates an execution of CONTRACTION.

Figure 4 here

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 $O(k)$ (Blum et al., 1973; Dor and Zwick, 1999). 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 20, $v(i_1), \ldots, v(i_{k-\delta}) \geq m_o$. 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 21 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:

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

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 $D'_j:\ v'_j(i_1), \ldots, v'_j(i_\delta)$, and uses them to expand S^j and create a primer $P^j = p_1^j \ldots p_k^j$, as follows:

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

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

Denote by m_c and m_e the number of strings covered by the primers P^c and P^e, respectively. Lemma 22 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 5. 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.

Figure 5 here

Lemma 22 $m_e \geq m_c$.

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 $v(i_1), \ldots, v(i_{k-\delta})$ be the $k-\delta$ largest leading values in D. CONTRACTION sets

positions $i_1, \ldots, i_{k-\delta}$ 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 $c(i_1), \ldots, c(i_{k-\delta})$ 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: $\{1 \leq i \leq k \mid i \neq i_1, \ldots, i_{k-\delta}\}$. 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 23 EXPANSION *approximates MC-DPD* within a factor of* $(k - \delta)$ *in time* $O(n^2 k)$.

■

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 pre-defined integer, $1 \leq x \leq k - \delta$. For simplicity, assume $x \mid (k - \delta)$. Denote by $\bar{b} = (b_1, \ldots, b_x)$ a binary vector of length x, or x-tuple, and denote by $\bar{i} = (i_1, \ldots, i_x),\ 1 \leq i_j \leq k$, 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 $i_1, \ldots, i_x$ in the input strings, i.e.:

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

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, $P^x = p^x_1 \ldots p^x_k$, $p^x_j = \{0, 1\}$, 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 6. 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 (Keich and Pevzner, 2002). Figure 6 here

Theorem 24 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 $i_1, \ldots, i_x$. P^o does not match u_o input strings, hence:

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

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 $\lceil \frac{k-\delta}{x} \rceil u_o$.

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 19).

Proposition 25 *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 section, we describe heuristics for MC-DPD with non-binary alphabets that are based on CONTRACTION and EXPANSION, and perform well in practice.

5 Implementation: The HYDEN Program

We developed and implemented an efficient heuristic, called HYDEN (Linhart and Shamir, 2003), 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 7. 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 is freely available for academic use (http://www.math.tau.ac.il/∼rshamir/hyden/HYDEN.htm).

Figure 7 here

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 $\Sigma = \{\text{A,C,G,T}\}$ 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 8). 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.

Figure 8 here

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 9. 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 10. 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 $O(N_a kL)$.

Figures 9 and 10 here

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 11). 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 $O(N_g\bar{r}k^3L)$.

Figure 11 here

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 section, 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 section.

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.

6 Applications

6.1 Deciphering the Human Olfactory Subgenome

HYDEN was originally developed and implemented as part of DEFOG, an experimental scheme for DEciphering Families Of Genes (Fuchs et al., 2002). 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. DEFOG consists of several computational and experimental phases. First, given a subset of known gene sequences, HYDEN is used to design degenerate primer pairs. The primers are then used in PCR to amplify fragments of genes, known as well as unknown, of the same family. The fragments are cloned, and an oligofingerprinting (OFP) process (Clark et al., 1999; Herwig et al., 2000; Meier-Ewert et al., 1998; Radelof et al., 1998) characterizes the clones by their patterns of hybridization with a series of very short (8-mer) oligonucleotides. Another novel algorithm, called CLICK (Sharan and Shamir, 2000; Sharan et al., 2003), clusters the clones into groups corresponding to the same gene according to their hybridization patterns. Finally, representatives from each cluster are sequenced and compared to the known gene sequences. The DEFOG project is joint work with the groups of H. Lehrach (MPI Berlin) and D. Lancet (Weizmann).

The DEFOG scheme was applied to the human olfactory receptor (OR, in short) subgenome. The human genome contains more than 1000 OR genes, of which more than 60% are considered pseudogenes (Glusman et al., 2001; Zozulya et al., 2001). OR genes have a single coding exon of about 1Kbp, and code for seven-transmembrane domain proteins (Buck and Axel, 1991). They have several highly conserved regions, primarily in transmembrane (TM) segments 2 and 7. In contrast, TM segments 4 and 5 show a high degree of variability — a crucial feature for recognizing a huge variety of odorants (Pilpel and Lancet, 1999).

Our experiment began with an initial collection of 127 OR genes, whose full DNA coding sequences of size 1Kbp were known at the time (Fuchs et al., 2000). This collection comprised our training set, on which HYDEN designed the primers. 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 — 6 for the 5' side (denoted L5, L9, L10, L20, L31 and L131) and 7 for the 3' side (R5, R20, R28, R73, R110, R147 and R442), of lengths $k = 26, 27$ and various degeneracies between $4,608$ and $442,368$ (the primers were named Dn, where D is 'L' for 5' and

'R' for 3', and n is the rounded degeneracy of the primer in thousands). 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. 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 1), and used them in PCR reactions. The degeneracy of a pair of primers 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 — data not shown).

Table 1 here

Table 1 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 1 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 12 shows the sequencing efficacy of several of the primer pairs we used, as a function of the degeneracy.

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. 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 12, 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 clones we sequenced include 140 clones from six clusters, which we sequenced exhaustively in order to obtain statistics on the quality of the clustering analysis (see (Fuchs et al., 2002)). The reported sequencing efficacy is therefore lower than the true efficacy of the primers.

Figure 12 here

The DEFOG experiment almost tripled the size of our initial OR repertoire, from 127 genes 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 (Fuchs et al., 2002).

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 (Glusman et al., 2001) [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. Figure 13 shows the 3-mismatches coverage of several primer pairs, both for the

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

training set and the test set (Table 1 contains the full data). Again, in order to allow a fair comparison between the primers, we included only the pairs L5/R5, L10/R5, L5/R28, L10/R28, L31/R73, and L31/R442, as well as a couple of additional primers that were designed by HYDEN but were not used in the experiment. Figure 13 here

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% 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.

6.2 The Canine Olfactory Subgenome

Encouraged by the results we obtained for the human OR subgenome, we launched a project with the group of D. Lancet (Weizmann) for analyzing the canine OR subgenome. We used two approaches: data mining in the Celera 1.3X sequence coverage of the dog genome, and a simplified version of DEFOG, in which we skipped the OFP and clustering phases (i.e., clones were selected for sequencing arbitrarily, rather than based on the fingerprints clustering). 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 $1.2 \cdot 10^6$ 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. The 1200 clones sequenced contained 246 distinct canine OR genes (the full dog OR repertoire is estimated to contain some 1200 genes). About 14% of these genes are pseudogenes, similar to the ratio in mouse (20%) (Young et al., 2002; Zhang and Firestein, 2002), but far from the ratio in human ($> 60\%$) (Glusman et al., 2001; Zozulya et al., 2001). 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 (Olender et al., 2004). This project demonstrates that DEFOG can be applied to study an unsequenced genome using degenerate primers designed according to a related species.

6.3 Olfaction vs. Vision among Primates

Another interesting project that utilized HYDEN is described in (Gilad et al., 2004). In that study, degenerate primer pairs designed based on human ORs were used to sequence 100 OR genes in human and in 18 primate species, for which the genome sequence is not available, including apes, Old World monkeys (OWMs) and New World monkeys (NWMs). As expected, the proportion of OR pseudogenes in human was found to be very high (above 50%). In great apes and OWMs, roughly 30% of the sequenced ORs are pseudogenes, whereas in NWMs this ratio is significantly lower — only 18% are pseudogenes. However, there is one exception: one NWM species, the howler monkey, was found to have a similar proportion of OR pseudogenes (31%) to that of OWMs and apes. Gilad et al. noticed that another phenotype that is shared only by the howler monkey, OWMs, and apes is full trichromatic color vision. Thus, the deterioration of the olfactory subgenome repertoire and the acquisition of full trichromatic vision occurred independently in two separate evolutionary branches: in the common ancestor of OWMs and apes, and in the New World howler monkey. This suggests an association between two senses on an evolutionary genetic scale: as vision improved in some of the primate species, they became less dependent on their sense of smell, which led to its decline.

7 Concluding Remarks

In this work, 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.

On the theoretical side, one may wish to design approximation algorithms for MC-DPD with 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 (Kwok et al., 1994), and that different types of mismatches have different effects on the reaction, e.g., A:C is less disruptive than A:G (Kwok et al., 1990). 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.

We have recently extended HYDEN, so that it could design several primer pairs. The first pair is constructed by running the algorithm described in Section 5 twice — for designing primers on the 5' and on the 3' side of the DNA sequences (the distance between the two regions can be set according to the specific requirements of the experiment). After the first primer pair is selected, all matching sequences are removed, and a second pair is designed using the remaining sequences. We repeat this process until a sufficiently large fraction of the input sequences is covered by the primers. This iterative procedure, described independently in (Souvenir et al., 2003), is a heuristic for solving MP-DPD. It is useful when more than one primer pair is required in order to

reach satisfactory coverage. Another heuristic for solving MP-DPD is the program MIPS, which was reported to outperform the iterative version of HYDEN when applied to the task of designing multiplex PCR experiments for SNP genotyping (Souvenir et al., 2003). As noted by Souvenir et al., the problems solved by the two algorithms are quite different — mainly, MIPS constructs a set of primers for one PCR experiment with multiple primers, whereas HYDEN designs primer pairs for separate experiments (one pair per experiment). If one wishes to use HYDEN for multiplex PCR, a better approach would be to design a set of 5' primers and a set of 3' primers separately. Each set could be constructed using an iterative procedure similar to the one described above (but on one side only), until sufficient coverage is reached. It would be interesting to compare the performance of this version of HYDEN to that of MIPS. Note that when using several different primers in the same PCR, one has to make sure the primers will not hybridize with one another. Both MIPS and HYDEN ignore this crucial issue, so additional tools should be used to check whether the designed primers might cross-hybridize.

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 (Thompson et al., 1994) or BlockMaker (Henikoff et al., 1995). 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. We are currently involved in several projects that use degenerate primers to study gene families and cDNA libraries. HYDEN is also being employed by several other labs for various tasks.

Acknowledgments

We 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), Roded Sharan and Dmitry Shmulevich (Tel-Aviv University). We thank David Johnson for his remarks on the Bin Packing problem. This work was partially supported by GIF grant G–0506–183.0396 and by the Israel Science Foundation (grant 309/02).

References

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

Blum, M., Floyd, R., Pratt, V., Rivest, R., and Tarjan, R. (1973). Time bounds for selection. *Journal of Computer and System Sciences*, 7:448–461.

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

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

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

Doi, K. and Imai, H. (1997). 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.

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

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

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

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco.

Gilad, Y., Wiebe, V., Przeworski, M., Lancet, D., and Pääbo, S. (2004). Loss of olfactory receptor genes coincides with the acquisition of full trichromatic vision in primates. *PLoS Biology*, 2(1):E5.

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

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

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

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

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

Johnson, D. (2002). Private communication.

Karp, R. (1972). Reducibility among combinatorial problems. In Miller, R. and Thatcher, J., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New-York.

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

Kwok, S., Chang, S., Sninsky, J., and Wang, A. (1994). A guide to the design and use of mismatched and degenerate primers. *PCR Methods and Appl.*, 3:S39–47.

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

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

Linhart, C. (2002). The degenerate primer design problem. Masters thesis, School of Computer Science, Tel Aviv University, November 2002, `http://www.math.tau.ac.il/~chaiml/biology/dpd_thesis.ps.gz`.

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

Linhart, C. and Shamir, R. (2003). HYDEN – A software for designing degenerate primers: `http://www.math.tau.ac.il/~rshamir/hyden/HYDEN.htm`.

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

Mullis, K., Faloona, F., Scharf, S., Saiki, R., Horn, G., and Erlich, H. (1986). Specific enzymatic amplification of dna in vitro: the polymerase chain reaction. *Cold Spring Harbor Symposium in Quantitative Biology*, 51:263–273.

NC-IUB (1985). Nomenclature for incompletely specified bases in nucleic acid sequences — recommendations 1984. *Biochemical Journal*, 229:281–286.

Olender, T., Fuchs, T., Linhart, C., Shamir, R., Adams, M., Kalush, F., Khen, M., and Lancet, D. (2004). The canine olfactory subgenome. *Genomics*, 83(3):361–372.

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

Pevzner, P. and Sze, S. (2000). 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.

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

Radelof, U., Hennig, S., Seranski, P., Steinfath, M., Ramser, J., Reinhardt, R., Poustka, A., Francis, F., and Lehrach, H. (1998). 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.

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

Sharan, R., Maron-Katz, A., and Shamir, R. (2003). CLICK and EXPANDER: A system for clustering and visualizing gene expression data. *Bioinformatics*, 19(14):1787–1799.

Sharan, R. and Shamir, R. (2000). 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.

Souvenir, R., Buhler, J., Stormo, G., and Zhang, W. (2003). Selecting degenerate multiplex PCR primers. In *Proc. 3rd Workshop on Algorithms in Bioinformatics (WABI 2003)*, pages 512–526.

Thompson, J., Higgins, D., and Gibson, T. (1994). CLUSTALW: 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.

Vishnevsky, O., Podkolodnaya, O., and Babenko, V. (1998). 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.

Wei, X., Kuhn, D., and Narasimhan, G. (2003). Degenerate primer design via clustering. In *Proc. 2nd IEEE Computer Society Bioinformatics Conference (CSB 2003)*, pages 75–83.

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

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

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

Table 1: 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 Figure 13.

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 1

Figure 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.

Figure 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 ($\{0, 1\}$).

Figure 3: Illustration of the reduction from MBP to MP-DPD.

Figure 4: 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.

Figure 5: Illustration of the first two iterations of EXPANSION on the eight strings from Figure 4. 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.

Figure 6: 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.

Figure 7: The HYDEN algorithm.

Figure 8: The basic alignment phase in HYDEN.

Figure 9: The H-CONTRACTION algorithm used by HYDEN.

Figure 10: The H-EXPANSION algorithm used by HYDEN.

Figure 11: The greedy hill-climbing procedure used by HYDEN. $m(P)$ denotes the coverage of primer P.

Figure 12: Sequencing efficacy of several primer pairs in the DEFOG experiment. The dotted line shows the percent of new genes, i.e., genes that were not in the training set, out of all the sequenced clones.

Figure 13: 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.

The Degenerate Primer Design Problem

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

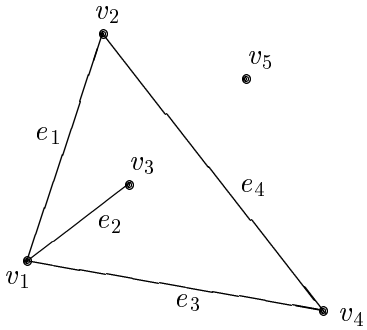
S_1 =TCGGC**TTGCAAG**CGTACT
S_2 =GGC**TTCCAGG**TCTTATAAGTC
S_3 =GCTTCCACGGTGCGAATCAGGGCTG
S_4 =A**TTGCTAG**GTTCAGGTA
S_5 =GCAAGGTATC**TTGCCAG**CTTTGA

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

Figure 1: (Linhart & Shamir)

CLIQUE **Minimum Coverage DPD**

Input: Graph $\mathcal{G} = (V, E)$, $|V| = 5$, $|E| = 4$, $c = 3$



$\Longrightarrow$

Input: $n = 4$, $k = 5$, $d = 2^3$, $m = \binom{3}{2} = 3$

$S_{e_1} = 11000$

$S_{e_2} = 10100$

$S_{e_3} = 10010$

$S_{e_4} = 01010$

$\Downarrow$

Solution: Clique $= \{v_1, v_2, v_4\}$ $\Longleftarrow$ *Solution:* $P = **0*0$

Figure 2: (Linhart & Shamir)

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: (Linhart & Shamir)

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:

4	2	1	**6**	0	5	3	**7**	4
4	**6**	**7**	2	**8**	3	5	1	4

$\Downarrow$

Output:

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

Figure 4: (Linhart & Shamir)

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

S^1: 011010101

S^2: 010010000 $\Longrightarrow$

S^3: 111010100

. . . (as in Figure 4)

Column distribution matrix D:

4	2	1	6	0	5	3	7	4
4	6	7	2	8	3	5	1	4

$\Downarrow$

Starting string: S^1 $\Rightarrow$

D':	0	2	1	0	0	0	**3**	0	**4**
	4	0	0	2	0	**3**	0	1	0

$\Downarrow$

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

Starting string: S^2 $\Rightarrow$

D':	0	2	0	0	0	0	0	0	0
	4	0	**7**	2	0	3	**5**	1	**4**

$\Downarrow$

P^2:	$*$	1	$*$	0	1	0	$*$	0	$*$

Figure 5: (Linhart & Shamir)

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

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

$\Longrightarrow$

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 * * 0 * * *

$\Downarrow$ Iteration 2

Output: P^x : 1 * 0 0 * 0 *

Figure 6: (Linhart & Shamir)

HYDEN ($I = \{S^1, \ldots, S^n; k; d; e\}$)*:*

Phase 1: $A_1, \ldots, A_{N_a} \leftarrow$ H-Align(I).

Phase 2: Foreach alignment A_i, $i = 1, \ldots, N_a$ do:

$P_i^c \leftarrow$ H-Contraction($I; A_i$).

$P_i^e \leftarrow$ H-Expansion($I; A_i$).

Sort primers $\{P_i^c, P_i^e \mid i = 1, \ldots, N_a\}$ 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 7: (Linhart & Shamir)

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 8: (Linhart & Shamir)

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 9: (Linhart & Shamir)

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 10: (Linhart & Shamir)

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 11: (Linhart & Shamir)

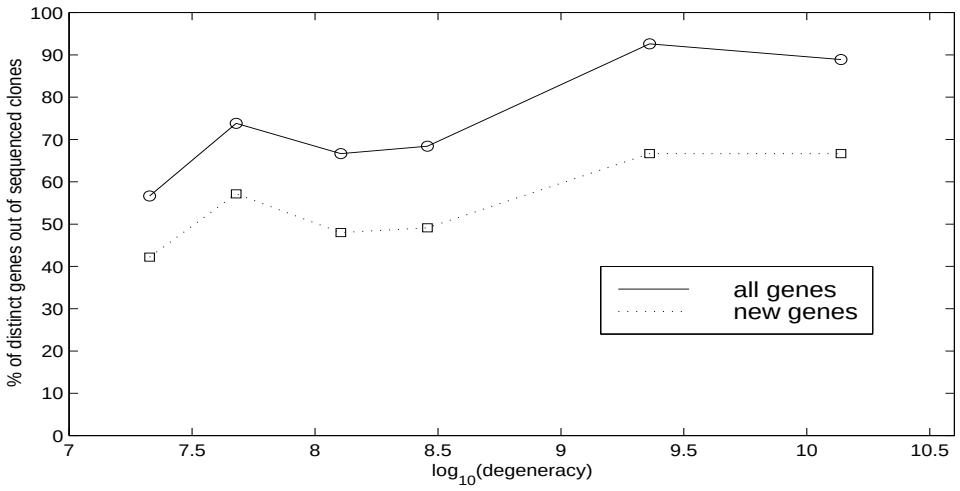


Figure 12: (Linhart & Shamir)

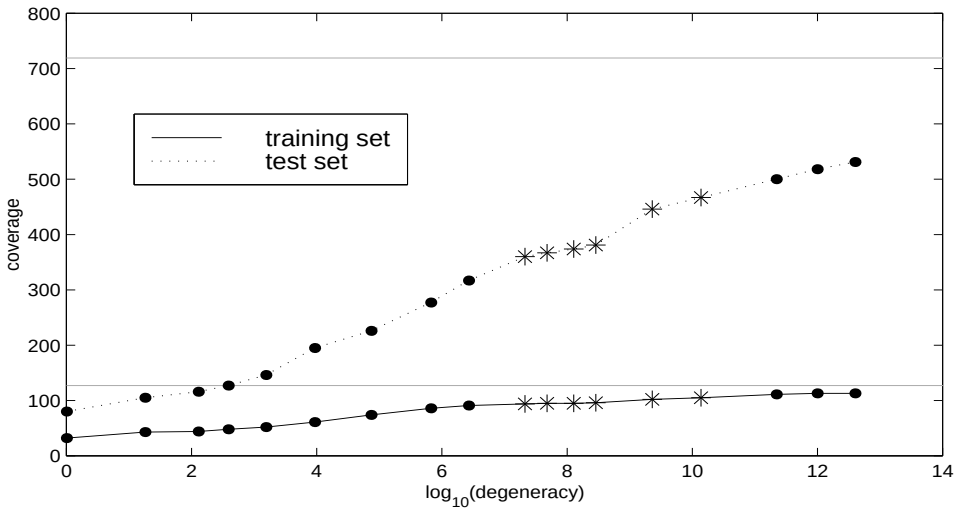


Figure 13: (Linhart & Shamir)