Chapter 1

Degenerate Primer Design: Theoretical Analysis and the HYDEN program[1]

Non-standard format chapter

Chaim Linhart and Ron Shamir

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, for example, when studying a family of genes that is known only in part, or is known in a related species. We discuss the complexity of several versions of the problem, and give approximation algorithms for one simplified variant. Based on these algorithms, we developed a program called HYDEN for designing highly-degenerate primers for a set of genomic sequences. We describe HYDEN, and report on its success in several applications for identifying olfactory receptor genes in mammals.

Keywords: Degenerate Primers for PCR, DPD, HYDEN, Olfactory Receptor Genes.

1 Introduction

A *degenerate* PCR primer is a primer sequence that contains several possible bases in one or more positions [1]. 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 total number of 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

[1]This work was partially supported by the German-Israeli Foundation for Scientific Research (G.I.F.) under grant G-0506-183.0396, and by the Israel Science Foundation (grant 309/02).

use low degeneracy of up to hundreds. In this chapter we study the problem of designing primers of high degeneracy from the theoretical and practical perspectives.

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, as well as perhaps additional related sequences that are unknown yet. 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, and specificity will decrease. We may have to compromise by aiming to match many but not necessarily all the sequences. We describe here 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). In the next sections, we define and analyze several variants of DPD, and describe a program we developed, called HYDEN, for producing high-degeneracy primers. Finally, we report results of several projects that used degenerate primers for amplifying olfactory receptor (OR) genes in various mammals. The theoretical results have been described in detail in [2], and are reprinted with permission. The experimental results are based on [3].

1.1 Related Problems

DPD is related to the Primer Selection Problem (PSP) [4], 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., [5]). 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 related genes. Also, in contrast to a single pair of degenerate primers for DPD, here the cost of generating the primers depends on the set size.

Since a degenerate primer can be viewed as a motif, DPD is also related to motif finding. However, there are marked differences: Motif finding algorithms (e.g., MEME [6], Gibbs Sampler [7]) 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, 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.

2 Theoretical Analysis

In this section we formally define several versions of DPD, report on hardness results, and briefly describe approximation algorithms for one key variant of the problem. The full proofs are given in [8, 2]. For basic background on algorithms and complexity we refer the reader to [9].

2.1 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 genes, one should obviously use highly degenerate primers. On the other hand, in order to reduce the chance of amplifying non-related sequences, the degeneracy must be bounded.

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., $P = p_1 p_2 \dots p_k$, 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 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 IUPAC nucleotide code: $P^* = \text{ASNGT}$. Let $\delta(P)$ be the number of degenerate positions in P. Clearly, $\lceil \log_{|\Sigma|} d(P) \rceil \leq \delta(P) \leq \lfloor \log_2 d(P) \rfloor$. A primer $P^1 = p_1^1 p_2^1 \dots p_k^1$ is a *sub-primer* of a primer $P^2 = p_1^2 p_2^2 \dots p_k^2$ of the same length, if $\forall i, 1 \leq i \leq k,\ p_i^1 \subseteq p_i^2$. 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 $P = p_1 p_2 \dots p_k$ *matches* a string $S = s_1 s_2 \dots s_l$, $s_i \in \Sigma$, 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?*

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 explain 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:

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.*

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 internal mismatches [1]. 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) per string.*

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 [10].

The real problem of designing degenerate primers requires the construction of one or more *pairs* of primers, so that each of the given genes matches at least one of the primer pairs with only a few mismatches. For an effective PCR we should require that the distance between the 5'- and the 3'-primer match site is large enough (i.e., the amplified region is sufficiently long for biological study). Other factors that influence PCR may also be incorporated, such as the positions of the mismatches and the GC content [1]. Our theoretical results focus on the simple, restricted DPD variants. As we shall now see, even those are hard.

2.2 Complexity Results

Using exhaustive search algorithms, it is possible to solve restricted cases of DPD in polynomial time. For example, if $d = O(1)$[2], we could consider all $< L$ substrings, where L is the sum of the lengths of the input strings, 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 $O(kL^2\binom{k}{\delta}\sigma^\delta)$. We shall later introduce an efficient approximation algorithm that is a variant of this exhaustive search.

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 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 $O(kLL^d)$. Theorem 6 summarizes restricted cases of DPD that can be solved in polynomial time [2].

Theorem 6 *DPD is polynomial when* $d = O(1)$*, or* $m = O(1)$*, or* $k = O(\log L)$.

[2] See [9] for the definition of the Oh notation.

Unfortunately, all the versions of DPD we defined are, in the general case, difficult problems [8]:

Theorem 7 *The following problems are $\mathcal{NP}$-Complete: MC-DPD (for $|\Sigma| \geq 2$), MD-DPD (for $|\Sigma| \geq 3$), MD-EDPD (for $|\Sigma| \geq 2$, even if $e = 1$ and all input strings are of length k), and MP-DPD (for $|\Sigma| \geq 2$).*

Furthermore, in MD-DPD and MD-EDPD it is difficult to approximate the *number* of degenerate positions in an optimal primer [8]:

Theorem 8 *Assuming $\mathcal{P} \neq \mathcal{NP}$, there is no polynomial time algorithm that approximates the number of degenerate positions in: (a) MD-DPD, within a factor of $c \cdot \log n$, for some constant $c > 0$; (b) MD-EDPD, within a factor of 1.36, even when $e = 1$ and all strings are of length k.*

3 Approximation Algorithms

In this section we describe several polynomial approximation algorithms for MC-DPD over the binary alphabet — $\Sigma = \{0, 1\}$. In this case, the number of degenerate positions in a primer is always $\delta(P) = \log_2 d(P)$.

3.1 Simple Approximations

Denote by $M(P)$ the set of input strings matched by a primer P. Let P^o be an optimal solution with degeneracy d to an instance of MC-DPD. Like any other primer with degeneracy d, P^o is a union of d non-degenerate primers (strings of length k): $P^o = \bigcup_{i=1}^{d} P^i$, where $P^1,\ldots,P^d$ constitute *all* the non-degenerate sub-primers of P^o, and $M(P^o) = \bigcup_{i=1}^{d} M(P^i)$. Let P^m be a sub-primer with the largest coverage, i.e., $|M(P^m)| = \max_{i=1}^{d}\{|M(P^i)|\}$. Then, obviously, $|M(P^o)| \leq d \cdot |M(P^m)|$. It is now clear how one can obtain a d-approximation to P: Simply traverse all k-long substrings of the input strings, and choose a substring P_0 that matches a maximum number of input strings. Since $|M(P^m)| \leq |M(P_0)|$, we get: $|M(P_0)| \geq |M(P^o)|/d$. The algorithm runs in time $O(kL^2)$ $(= O(k^3n^2)$, since 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 now describe another algorithm, which starts with a completely degenerate primer, and gradually refines, or "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 9 *MC-DPD can be approximated within a factor of* $2^{k/2}$ *in time* $O(kL)$.

3.2 Approximating the Number of Unmatched Strings

Unlike the previous algorithms we studied, we shall now describe several algorithms that approximate the number of *unmatched* strings. In other words, we now treat MC-DPD 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. 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 first two 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, i.e., $u_o = n - m_o$. Clearly, $\forall b \notin p_i^o$, $D(b,i) \le u_o$, and for each non-degenerate position i in P^o, $D(p_i^o, i) \ge m_o$. Since P^o contains $k-\delta$ non-degenerate positions, it follows that there are $k-\delta$ (or more) columns in D with a value at least m_o. Given a column distribution matrix D, we define the *leading value* of column i, denoted $v(i)$, as the largest value in that column: $v(i) = \max\{D(b,i) \mid b \in \Sigma\}$. Similarly, the *leading character* of column i is a character $c(i)$, whose count is the leading value: $D(c(i), i) = v(i)$. Let $v(i_1) \ge v(i_2) \ge \ldots \ge v(i_k)$ be the leading values in D, sorted from largest to smallest. The following lemma follows from the discussion above.

Lemma 10 *If P^o covers m_o strings, then $v(i_{k-\delta}) \geq m_o$.*

The CONTRACTION Algorithm

The CONTRACTION 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. 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 earlier — CONTRACTION uses the column distribution matrix to guide it in selecting good positions to refine, instead of choosing them arbitrarily. Figure 1.1 illustrates an execution of CONTRACTION. *Figure 1.1 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 $O(k)$ [11]. At each degenerate position, the primer P^c has no mismatches with the input strings. According to Lemma 10, 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 11 CONTRACTION *approximates MC-DPD* within a factor of $(k-\delta)$ in time $O(nk)$.*

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 $P^j = p_1^j \dots p_k^j$, as follows:

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

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

Denote by m_c and m_e the number of strings covered by the primers P^c and P^e, respectively. It is possible to show that $m_e \geq m_c$ [2], which implies that EXPANSION also guarantees a $(k-\delta)$-approximation to MC-DPD*. In fact, in some cases EXPANSION may find a better primer than CONTRACTION, as demonstrated in Figure 1.2. On the down side, EXPANSION is slower — its running time is $O(n^2 k)$, dominated by the coverage computation of the n primers it constructs.

Figure 1.2 here

Corollary 12 EXPANSION *approximates MC-DPD* within a factor of $(k-\delta)$ in time $O(n^2 k)$.*

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 $\bar{b} = (b_1, \dots, b_x)$ a binary vector of length x, or x-tuple, and denote by $\bar{i} = (i_1, \dots, i_x)$, $1 \leq i_j \leq k$, a set of x distinct positions. Define the *multi-column distribution matrix* $MD(\bar{b}, \bar{i})$ as the count of the x bits of $\bar{b}$ at positions $i_1, \dots, i_x$ in the input strings, i.e.:

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

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, $P^x = p_1^x \dots p_k^x$, $p_j^x = \{0,1\}$, 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: $\forall 1 \leq j \leq x \ , \ p_{i'_j}^x = b'_j$. 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 $r = (k-\delta) \bmod x$, 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. The performance of CONTRACTION-X is summarized in the following theorem [2]:

Theorem 13 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* $O(\binom{k}{x} nx)$.

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 [12].

3.3 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 3.1 are easily generalized to larger alphabets, as follows. 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 $O(kL)$. 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 9).

Proposition 14 *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 3.2 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 among 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.

4 The HYDEN Program

We developed and implemented an efficient heuristic, called HYDEN [13], for designing highly degenerate primers. The input to HYDEN is a list of DNA sequences and a set of integers that control the number of primer pairs to design, the length and maximum degeneracy of the primers, and other parameters of the algorithm. HYDEN constructs primer pairs with the specified length and degeneracy that together cover many of the given sequences. Each primer is designed by running a 3-phase algorithm, outlined in Figure 1.3. 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. This procedure is repeated to design several pairs of 5' and 3' primers, as explained in Section 4.2. HYDEN is written in C++, and runs under Windows and Linux. HYDEN is freely available for academic use (http://www.cs.tau.ac.il/∼rshamir/hyden).

Figure 1.3 here

4.1 The HYDEN Algorithm

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 1.4). Let $T = t_1t_2\ldots t_k$ 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 1.4 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 1.5. 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 1.6. Notice that the original EXPANSION algorithm repeats this procedure for each substring in A. However, early experiments demonstrated that in many cases, there is little difference between primers obtained by expanding different substrings in A. 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 $O(N_a k L)$.

Figures 1.5 and 1.6 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 1.7). 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 $O(N_g \bar{r} k^3 L)$.

Figure 1.7 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. 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 Section 5.

HYDEN is a generalization of the $(k - \delta)$-approximation of MC-DPD* that we presented in Section 3.2. 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.

4.2 Designing Multiple Primer Pairs

The HYDEN program can design several primer pairs. The first pair is constructed by running the algorithm we have just described 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 according to the number of primer pairs requested by the user. This iterative procedure, described independently in [10], 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 HYDEN when applied to the task of designing multiplex PCR experiments for SNP genotyping [10]. 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, since each 5' primer may pair with each 3' primer, whereas HYDEN constructs one 3' primer per 5' primer. 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.

4.3 Running HYDEN

HYDEN receives an input file that contains a set of DNA sequences in Fasta format, and a list of command-line parameters that specify the number of primers to design, their length and degeneracy, the regions within the sequences to use for designing the 5' and 3' primers, the maximal number of mismatches to allow between the primers and the sequences they match, and the parameters N_a, N_g and N_h ($N_{a'}$ is automatically set to $5N_a$). For a more detailed description, see the "Readme.txt" file supplied with HYDEN.

HYDEN is distributed with a sample input file, called "HGP_50genes.fasta", which contains 50 human olfactory receptor genes of length roughly 1Kbp. Figure 1.8A demonstrates an execution of HYDEN on these sequences. Given the parameters shown in the figure, HYDEN designs two pairs of primers. The first 300bp (last 350bp) in each sequence are used for designing the 5' (3') primers. The primers are of length 25, and maximum degeneracy 5,000 (5') or 30,000

(3'). Each primer is allowed to have up to two mismatches with each sequence it covers, and a total of three mismatches are allowed between a pair of primers and each covered sequence. HYDEN reports the progress of the algorithm, and summarizes its results in a table. Figure 1.8B shows the summary table from HYDEN's output on the sample data. The first primer pair matches 34 (68%) of the 50 input sequences. Together, the two primer pairs HYDEN designed cover 40 (80%) genes.

Figure 1.8 here

5 Applications

5.1 Deciphering the Human Olfactory Subgenome

HYDEN was originally developed and implemented as part of DEFOG, an experimental scheme for DEciphering Families Of Genes [3]. 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. 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 [14] characterizes the clones by their patterns of hybridization with a series of very short (8-mer) oligonucleotides. Another algorithm, called CLICK [15], 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 methodology was developed jointly with the groups of H. Lehrach (MPI Berlin) and D. Lancet (Weizmann).

The DEFOG scheme was applied to the human olfactory receptor (OR) subgenome. The human genome contains more than 1,000 OR genes, of which more than 60% are considered pseudogenes [16, 17]. OR genes have a single coding exon of about 1Kbp, and code for seven-transmembrane domain proteins [18]. 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 [19].

Our experiment began with an initial collection of 127 OR genes, whose full DNA coding sequences of size 1Kbp were known at the time [20] (the project began before the completion of the Human Genome draft sequence). This collection comprised our *training set*, on which HYDEN designed the primers. Altogether, we designed 13 primers — 6 for the 5' side, and 7 for the 3' side, of lengths $k = 26, 27$ and various degeneracies between 4,608 and 442,368. The primers on each side are quite similar to one another, and differ mainly in their degeneracy, except for four

special primers — one pair was designed at different positions, closer to the 5' and 3' ends of the genes, and another pair 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 $N_h = 50$, $N_a = 3{,}000$, $N_g = 100$), takes less than 10 minutes, distributed evenly among the three phases of the program, on a 1.5GHz Pentium4 PC.

We selected 20 different primer pairs from the 13 primers we designed, and used them in (separate) 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 — previous applications usually used 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).

We obtained a total of 13,580 clones from all the PCR experiments. We then applied the OFP process and the CLICK clustering software, which partitioned the clones into 239 clusters and 121 singletons (single clone clusters). We selected representative clones from each cluster, and successfully sequenced a total of 924 clones. The extremely degenerate primers we designed proved very effective: They achieved high sensitivity, amplifying a total of 300 unique OR genes, and extremely high specificity, yielding only 0.4% (4 out of 924) non-OR products. The *sequencing efficacy* of the primers, defined as the percentage of distinct genes that were obtained from each primer pair out of the total number of clones sequenced for that pair, was very high — for 10 out of 12 primer pairs with degeneracy over 10^9, sequencing efficacy was $79\% - 93\%$, and for all 8 pairs with lower degeneracy, it was $57\% - 79\%$. Another advantage of using highly degenerate primers is the large number of *new* genes they amplify — in our case, 231 out of 300 genes were new. Altogether, the DEFOG experiment almost tripled the size of our initial OR repertoire, from 127 genes to 358. The full experimental details and an analysis of the performance of the primers are reported in [2, 3].

After the publication of the first draft of the human genome, we analyzed the performance of various primer pairs on all full-length OR sequences that were computationally detected in the draft. This set consisted of 719 genes [16]. 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. Figure 1.9 shows the 3-mismatches coverage of several primer pairs, both for the training set and the test set. As expected, there is a 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. Second, as mentioned earlier, PCR gives a poor yield when the degeneracy is very high. Additional analyses of the performance of the primers in the DEFOG project are given in [2].

Figure 1.9 here

5.2 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 [21]. We used 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 1,200 clones we sequenced contained 246 distinct OR genes (the full dog OR repertoire is estimated to contain some 1,200 genes), again demonstrating the advantages of using highly degenerate primers for amplifying many related sequences. About 14% of the canine OR genes we obtained are pseudogenes, similar to the ratio in mouse (20%) [22, 23], but far from the ratio in human ($> 60\%$) [16, 17]. 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.

5.3 Olfaction vs. Vision among Primates

Another interesting project that utilized HYDEN is described in [24]. 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 was 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.

References

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

[2] C. Linhart and R. Shamir. The degenerate primer design problem: Theory and applications. *Journal of Computational Biology*, 12(4):431–456, 2005.

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

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

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

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

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

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

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, Mass., 1990.

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

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

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

[13] C. Linhart and R. Shamir, 2003. HYDEN – A software for designing degenerate primers: `http://www.cs.tau.ac.il/~rshamir/hyden`.

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

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

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

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

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

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

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

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

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

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

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

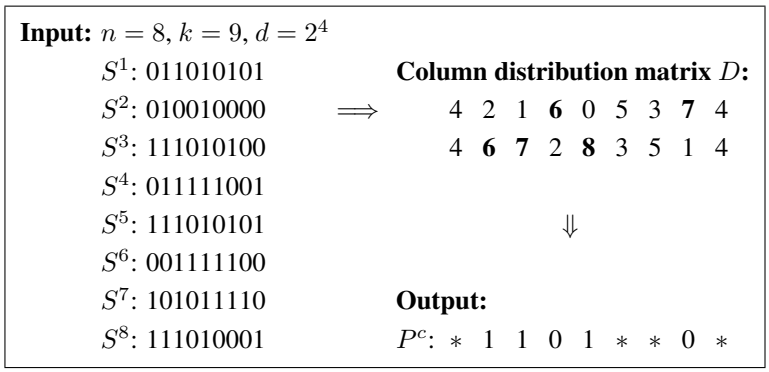

Figure 1.1: Example of an execution of CONTRACTION on eight strings. The five ($= k - \delta$) largest leading values in D are marked in bold face. The primer P^c covers four input strings — S^1, S^3, S^5 and S^8.

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

S^1: 011010101
S^2: 010010000
S^3: 111010100
. . . (as in Figure 1.1)

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 1.2: Illustration of the first two iterations of EXPANSION on the eight strings from Figure 1.1. The four ($= \delta$) largest leading values in D' are marked in bold face. The expansion of S^1 (P^1) covers four strings, and is identical to the primer constructed by CONTRACTION. The expansion of S^2 (P^2) covers five input strings — S^1, S^2, S^3, S^5, and S^8.

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 1.3: The HYDEN algorithm for designing a single primer.

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 1.4: The basic alignment phase in HYDEN.

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 1.5: The H-CONTRACTION algorithm used by HYDEN.

H-Expansion (I; A):

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

Figure 1.6: The H-EXPANSION algorithm used by HYDEN.

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

A. Command-line parameters:

```
hyden.exe -dna HGP_50genes.fasta -mprimers 2
          -from5  0 -to5  300 -len5 25 -deg5  5000 -mis5 2
          -from3 -1 -to3 -350 -len3 25 -deg3 30000 -mis3 2
          -mis 3 -nentropy 30 -nalgs 500 -nimprove 50
```

B. Output summary table:

Pair	5′ primer	3′ primer (inverted)	coverage	% (acc.)
0	CTNSAYDCNCCYATGTAYTTYTTHC	TCCTBADRSTRTARATNANNGGDTT	34	68 %
1	MCCCCATGTAYTTYTTYCYBDSMAN	ATKRCWGHYTTBAMHWCHTYATTYC	6	80 %

Figure 1.8: Example of an execution of HYDEN. (A) Command line for running HYDEN on the sample input file. (B) Output table of HYDEN, listing the two primer pairs it designed.

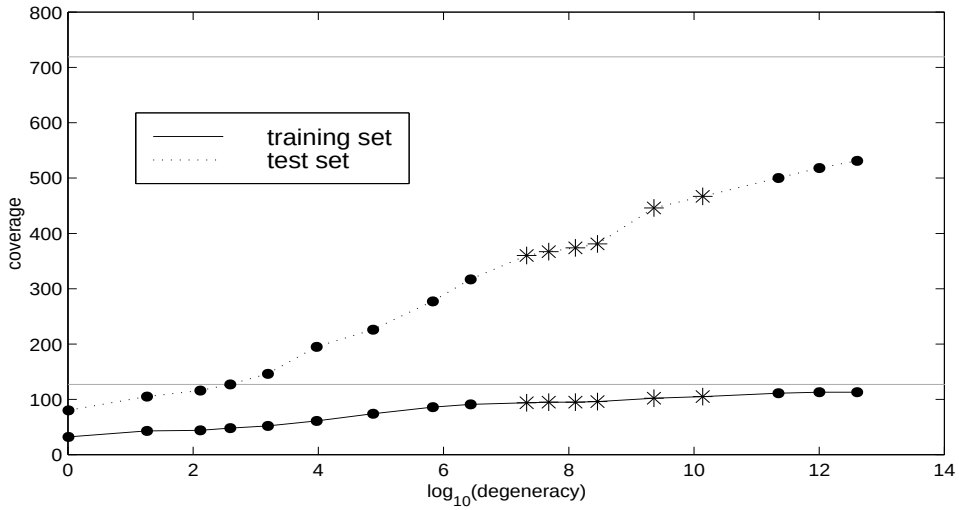

Figure 1.9: Training-set and test-set 3-mismatches coverage of primer pairs with various degeneracies in the human olfactory receptors project. Primers that were actually used in the DEFOG experiment are marked by asterisks. The horizontal lines mark the size of the training and test sets.