BIOINFORMATICS

Vol. 18 Suppl. 1 2002
Pages S172–S180

The degenerate primer design problem

Chaim Linhart and Ron Shamir

School Of Computer Science, Tel-Aviv University, Ramat-Aviv, Tel-Aviv, 69978, Israel

Received on January 24, 2002; revised and accepted on April 1, 2002

ABSTRACT

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

Availability: Available on request from the authors.

Contact: chaiml@tau.ac.il; rshamir@tau.ac.il

Keywords: PCR primer design; degenerate primers; optimization; human olfactory subgenome; protein families.

INTRODUCTION

A PCR (Polymerase Chain Reaction) primer sequence is called *degenerate* if some of its positions have several possible bases (Kwok *et al.*, 1994). For example, in the primer GG{CG}A{CTG}A the third position is C or G and the fifth is C, T or G. 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 degeneracies of up to hundreds. In this paper we study the problem of designing primers of high degeneracy.

In a typical situation, one has a collection of related target sequences, , 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 a solution is insufficient because of gaps, the inappropriate objective function of the alignment, and, most notably, the exceedingly high degeneracy: When degeneracy is too high, unrelated sequences may be amplified as well, losing specificity. We may have to compromise by aiming to match many but not necessarily all the sequences. Our goal here is to develop an *ad hoc* method for designing primers that will allow tradeoff between the degeneracy and the coverage (the number of matched input sequences). We call this objective Degenerate Primer Design (DPD).

There are various reasons for studying DPD. For example, we were involved in a project with the groups of H.Lehrach (MPI Berlin) and D.Lancet (Weizmann) for finding new human olfactory receptor (OR) genes. At the outset of the project (which preceded the publication of the human genome), only 127 OR genes were known, and the goal was to selectively amplify additional OR genes using degenerate primers. The rationale was that primers which match many of the known genes, would also amplify many new genes from the same family as well, whose sequences are closely related. Most OR genes contain conserved regions, and so the primers would be designed to match such regions. OR genes are a single 1000bp 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 (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 large portion of the genes without losing specificity is rather large. Furthermore, since the primers are not degenerate, they do not amplify many of the unknown genes. Traditionally, degenerate primers were usually designed manually by examining multiple alignments of the target sequences. CODEHOP is a program for designing degenerate primers (Rose *et al.*, 1998). For each given multiple alignment, it constructs a pair of primers. Each primer consists of a degenerate $3'$ core region and a $5'$ consensus sequence that stabilizes annealing. CODEHOP works well for small sets of proteins, taking into account the codon usage of the target genome, as well as the desired annealing temperature. However, it is inappropriate for constructing primers with very high degeneracy on large sets of long genomic sequences.

Since a degenerate primer can be viewed as a motif, DPD is also related to motif finding. However, there are marked differences: Motif algorithms (e.g., MEME (Bailey and Elkan, 1995), CONSENSUS (Hertz and Stromo, 1999), Gibbs Sampler (Lawrence *et al.*, 1993)) 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.

In this paper we study the DPD problem from theoretical and practical perspectives. From the theoretical point of view, we categorize several variants of the problem. In one key variant we bound the degeneracy and wish to maximize coverage, and in another we wish to minimize degeneracy while requiring full coverage. We give conditions under which the problem is polynomial, but prove that the two variants above and some others are in general $\mathcal{NP}$-Hard. For the maximum coverage variant, we provide several polynomial approximation algorithms. We then describe a practical program called HYDEN for producing high degeneracy primers. The program is a heuristic that builds on ideas analysed in the theoretical part. HYDEN was applied in the context of searching for new OR genes, where it designed primer pairs with degeneracy as high as $1.4 \cdot 10^{10}$. These 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 analyse 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).

The paper is organized as follows: We first give formal definitions of the problems. We then present hardness results, and describe polynomial algorithms for several problem variants, and approximation algorithms for more difficult variants. Finally, we introduce the HYDEN algorithm, and present the actual performance of HYDEN in the OR project. For lack of space some details and most proofs are omitted.

PROBLEM DEFINITION

Let Σ denote a finite fixed alphabet. In the case of DNA sequences, $\Sigma = \{A, C, G, T\}$. A *degenerate string* or *primer*, is a string P with several possible characters at each position, i.e., $P = p_1 p_2 \ldots p_k$, where $p_i \subseteq \Sigma$, $p_i \neq \emptyset$. k is the *length* of the primer. The number of possible character sets at a single position is $\sigma = 2^{|\Sigma|} - 1$. The *degeneracy* of P is $d(P) = \prod_{i=1}^{k} |p_i|$. For example, the primer $P^* =${A}{C,G}{A,C,T}{G}{T} is of length 5 and degeneracy $d(P^*) = 6$. At *non-degenerate positions*, i.e., positions that contain a single character, we shall often omit the brackets. 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 the primer $P^2 = p_1^2 p_2^2 \ldots p_k^2$, 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 sub-string that can be extracted from P by selecting a single character at each position, i.e., $\exists j, 0 \leq j \leq l - k$ such that $\forall i, 1 \leq i \leq k,\ s_{j+i} \in p_i$. 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?*

The value m is called the *coverage* of the primer. 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. k is fixed by PCR constraints to 20–30, so we focus on optimizing either the degeneracy or the coverage. Below are two simplified versions of those optimization problems.

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.*

The real problem of designing degenerate primers is more complicated in several ways. First, the sequence length is of course much longer than the primer's, and the allowed degeneracy does not admit complete coverage. Second, one should allow few mismatches between the primer and each input string. Third, two primers, $5'$ and $3'$, need to be designed together, with an appropriate distance between them. Fourth, we may have to construct several primer pairs, so that together they cover as many of the input strings as possible. We may have to incorporate additional 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.

COMPLEXITY

In this section we shall discuss the computational complexity of various variants of DPD. First, let us examine cases, for which we can suggest a polynomial solution. Suppose that $k = O(\log L)$, where L is the sum of the lengths of the input strings. A straightforward algorithm that checks all the $|\sigma|^k$ possible primers runs in time $O(kL|\sigma|^k)$. Note that primers are usually of length $k < 30$, but the obtained time bound is impractical. Now, 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 sub-string of one of the strings. More generally, if $d = O(1)$, we could consider all $O(L)$ k-long sub-strings, and try to increase the degeneracy of each candidate sub-string 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. At each degenerate position we could try all σ possible character sets. Thus, there is a total of fewer than $L\binom{k}{\delta}\sigma^\delta$ degenerate primers to check, and the total running time is $O(kL^2\binom{k}{\delta}\sigma^\delta)$. Another simple version of DPD is obtained when the number of strings the primer should match is bounded, i.e., $m = O(1)$. In this case, we could enumerate the m k-long sub-strings the primer may match. If their union is a primer with degeneracy d or less, then it is a valid solution. This algorithm runs in time $O(kL^m)$. In summary:

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

Another polynomial variant of DPD is obtained when we combine MC-DPD with MD-DPD, that is, given a set of strings of length k and an integer d, we wish to design a primer of length k and degeneracy at most d that matches all the input strings. A trivial polynomial algorithm is to simply compute the primer P, which is the union of all the input strings (i.e., in each position of P it consists of the set of characters that appear at that position in the strings). If $d(P) \le d$, then P is a feasible solution. Otherwise, there is no such solution. Interestingly, this polynomial variant of DPD regains its $\mathcal{NP}$-Completeness when we allow one mismatch between the primer and each string.

THEOREM 2. *DPD is polynomial when all strings are of length k and we require full coverage.*

The general DPD problem, as well as its simplified versions we defined in the previous section, are $\mathcal{NP}$-Complete. Due to lack of space, we omit the proofs.

THEOREM 3. *The following problems are $\mathcal{NP}$-Complete:*

(a) *MC-DPD (for $|\Sigma| \ge 2$).*

(b) *MD-DPD (for $|\Sigma| \ge 3$).*

(c) *The combination of MC-DPD with MD-DPD, when we allow one mismatch between the primer and each string it matches (for $|\Sigma| \ge 2$).*

One can also show strong inapproximability results for problems (b) and (c).

APPROXIMATION ALGORITHMS

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

Simple approximations

Denote by $M(P)$ the set of input strings matched by a primer P. If P is a union of two primers, P^1 and P^2, then clearly $M(P) = M(P^1) \cup M(P^2)$. Let P^o be an optimal solution 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): $P^o = \bigcup_{i=1}^{d} P^i$ ($P^1, \ldots, P^d$ 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, $|M(P^o)| \le d \cdot |M(P^m)|$. It is now clear how one can obtain a d-approximation to P: Simply find the non-degenerate primer P_0 that matches a maximum number of input strings. Since $|M(P^m)| \le |M(P_0)|$, we get: $|M(P_0)| \ge |M(P^o)|/d$. The algorithm runs in time $O(kL^2)$, where L is the sum of the lengths of the input strings. The running time can be reduced to $O(kL)$ using a hash table to store the number of strings matched by each sub-string. 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 ($1 \le \alpha \le \log_2 d$). P_α approximates MC-DPD within $d/2^\alpha$, since the optimal primer P^o can be represented as a union of $d/2^\alpha$ sub-primers, each one with degeneracy 2^α, such that 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 that starts with a completely degenerate primer and '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 $P_0^k = 0\{0,1\}\{0,1\}$ and $P_1^k = 1\{0,1\}\{0,1\}$. 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 the second position in the primer, i.e., replace it with either '0' or '1', whichever is better, and obtain a primer with degeneacy 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 $O((k - \delta)L)$.

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

COROLLARY 1. *MC-DPD can be approximated to within a factor $2^{k/2}$ in time $O(kL)$.*

Approximating the number of uncovered strings

In this section we describe two approximation algorithms—CONTRACTION and EXPANSION. Unlike the previous algorithms we studied, these algorithms approximate the number of *uncovered* 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 a 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.

Both algorithms construct the *column distribution matrix* $D(b, i)$ of 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 $D(b, i) = |\{j \mid s_i^j = b\}|$, $b \in \Sigma$, $1 \le i \le k$. 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, for each non-degenerate position i in P^o, $D(p_i^o, i) \ge m_o$. Equivalently, $\forall b \notin p_i^o$, $D(b, i) \le u_o$. 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: $v(i) = \max\{D(b, i) \mid b \in \Sigma\}$. 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 1. *If P^o covers m_o input strings, then $v(i_{k-\delta}) \ge m_o$.*

The first algorithm we describe is called CONTRACTION. The algorithm selects the $k - \delta$ largest leading values in D—$v(i_1), \ldots, v(i_{k-\delta})$, and sets the output primer P^c to contain the $k - \delta$ corresponding leading characters, and degeneracies at the rest of the positions, i.e.:

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

In a sense, this is a smart variation of the simple $2^{k-\delta}$-approximation algorithm we saw earlier. CONTRACTION starts with a completely degenerate primer, too, and uses the matrix D to guide it in selecting good positions to de-degenerate, instead of choosing them arbitrarily.

The running time of CONTRACTION is linear in the length of the input—$O(nk)$. 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 1, $v(i_1), \ldots, v(i_{k-\delta}) \ge m_o$. 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 4. *MC-DPD* can be approximated to within a factor $(k - \delta)$ in time $O(nk)$.*

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 $(b \in \{0, 1\}\ ,\ 1 \le i \le k)$:

$$D'_j(b, i) = \begin{cases} 0 & s^j_i = b \\ D(b, i) & \text{otherwise} \end{cases}$$

It then selects the δ largest leading values in D'_j—$v'_j(i_1), \dots, v'_j(i_\delta)$, and uses them to expand S^j and create a primer $P^j = p^j_1 \dots p^j_k$, as follows:

$$\forall 1 \le i \le k\ ,\quad p^j_i = \begin{cases} \{0, 1\} & i \in \{i_1, \dots, i_\delta\} \\ s^j_i & \text{otherwise} \end{cases}$$

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 2 establishes that P^e is at least as good as P^c, and, therefore, EXPANSION is also a $(k - \delta)$-approximation to MC-DPD*. In fact, as the lemma implies, in some cases EXPANSION may find a better primer than CONTRACTION (indeed, we can easily devise such examples). On the down side, EXPANSION is slower—its running time is $O(n^2k)$, dominated by the coverage computation of the n primers it examines.

LEMMA 2. *If $|\Sigma| = 2$, then $m_e \ge m_c$.*

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

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

Fig. 1. The HYDEN algorithm.

IMPLEMENTATION: THE HYDEN PROGRAM

We developed and implemented an efficient heuristic, called HYDEN, 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 1. In the first phase, HYDEN locates conserved regions in the DNA sequences by finding ungapped local alignments with a low entropy score. In the second phase, it designs primers using variants of the CONTRACTION and EXPANSION algorithms. Finally, it uses a greedy hill-climbing procedure to improve the primers, and selects the best one as the output.

Formally, let $I = \{S^1, \dots, S^n; k; d; e\}$ be the input to HYDEN, where $S^1, \dots, S^n$ are n strings over $\Sigma = \{A, C, G, T\}$ with a total length of L characters, and k, d, and e are the length, degeneracy, and mismatches parameters, respectively. Let N_a, $N_{a'}$, N_g and N_h be additional integer parameters, whose roles shall soon be described. Denote by A an ungapped local alignment (alignment, in short) of the input strings, that is, a set of n sub-strings of length k (actually, A is a multi-set, since it may contain several copies of a sub-string). Denote by D_A the column distribution matrix of the sub-strings 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 sub-strings in

A. The first phase of HYDEN exhaustively enumerates all sub-strings of length k in the input strings, and generates an alignment for each one, as follows. Let $T = t_1 t_2 \dots t_k$ be a sub-string of length k. In each input string S^j, HYDEN-Align finds the best match to T in terms of Hamming distance, i.e., the k-long sub-string T^j of S^j that has the smallest number of mismatched characters with T. The sub-strings $T^1, \dots, T^n$ (one of which is T itself) form the alignment A_T. After considering all $O(L)$ different sub-strings in the input, HYDEN-Align obtains $O(L)$ alignments. The N_a alignments with the lowest entropy score are passed to the second phase. HYDEN-Align runs in time $O(kL^2)$. Fortunately, a few simple heuristics can reduce the running time considerably with marginal impact on the quality of the results.

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 HYDEN-Align iterates all k-long sub-strings, and aligns only N_h input strings to each one. Then, the $N_{a'}$ sub-strings, whose alignments received the lowest (partial) entropy scores, are 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 HYDEN-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 sub-strings are very similar. Therefore, if the alignment we get from a sub-string $s_i \dots s_{i+k-1}$ has a high entropy score, there is no point in checking the next sub-string: $s_{i+1} \dots s_{i+k}$, as it is highly unlikely to yield good results, either. In fact, if the entropy score is very poor, we may decide to skip more than one sub-string. In practice, this simple idea reduced the running time of HYDEN-Align by a 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—*'HYDEN-Contraction'* and *'HYDEN-Expansion'*. These algorithms are generalizations of the CONTRACTION and EXPANSION approximation algorithms, respectively, to non-binary alphabets. HYDEN-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. HYDEN-Expansion employs an opposite approach. It uses the sub-string $T \in A$, 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. Notice that the original EXPANSION algorithm repeats this procedure for each sub-string in A. However, early experiments demonstrated that if most of the input strings can be covered by a single primer, there is very little difference between the primers that are obtained by expanding different sub-strings in A (data not shown). Therefore, in HYDEN-Expansion we chose to expand only one sub-string 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 $O(N_a kL)$.

The final phase of HYDEN tries to improve the N_g primers found in the previous phase using a simple hill-climbing procedure, called *'HYDEN-Greedy'*. Given a primer P, HYDEN-Greedy checks whether it can remove a character in a degenerate position in P and add a different character in any position instead, such that the coverage of the primer increases. This process is repeated as long as coverage is improving. Denote by r the number of iterations performed until a local maximum is reached. Then, the running time of HYDEN-Greedy is $O(rk^3L)$. In our experiments, r was almost always below 5. In order to limit the running time of HYDEN-Greedy 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 $O(N_g \bar{r} k^3 L)$.

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 performance of the program and the quality of the solution it provides. HYDEN is a generalization of the $(k - \lfloor \log_2 d \rfloor)$-approximation of MC-DPD* that we presented in the previous section. 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, 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, the results it produced in practice for the OR subgenome were highly satisfactory.

APPLICATION: DECIPHERING THE HUMAN OR SUBGENOME

HYDEN was studied and implemented as part of DEFOG—an experimental scheme for DEciphering Families Of

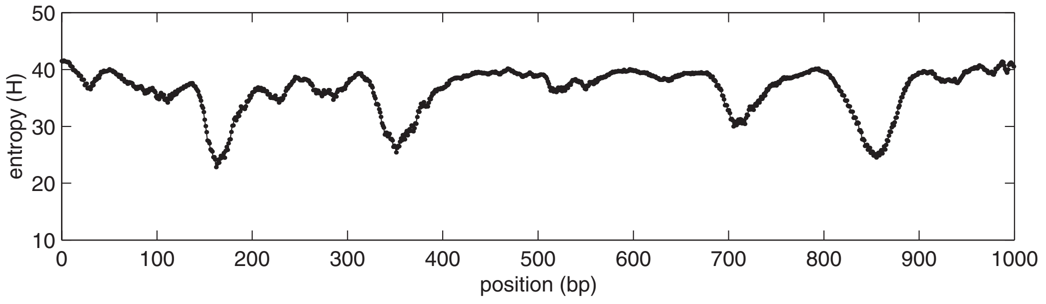

Fig. 2. Average entropy score along the 127 OR genes in the training set.

Genes. DEFOG provides a powerful means for analysing 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 analyse cDNA libraries of gene families. Given a subset of known gene sequences, HYDEN is used to design degenerate primer pairs. The primers are then used in PCR procedures to amplify fragments of known, as well as unknown, genes of the same family. The fragments are cloned, and an oligofingerprinting (OFP) process (Herwig *et al.*, 2000) characterizes the clones by their patterns of hybridization with a series of up to 200 short oligonucleotides. The hybridization pattern of a clone is called its *fingerprint*. Another novel algorithm, called CLICK (Sharan and Shamir, 2000), clusters the clones into groups corresponding to the same gene according to their fingerprints. Finally, representatives from each cluster are sequenced and compared to the existing database. DEFOG was implemented for the human olfactory receptor subgenome, which contains some 2–3% of all human genes (see, e.g., Lancet and Ben-Arie, 1993). In a single experiment, it almost tripled the size of our initial OR repertoire, from 127 genes to 358. The extremely degenerate primers we designed proved very effective, both in terms of sensitivity, amplifying 300 unique OR genes, and specificity, yielding only 0.4% non-OR products. The combination of the OFP process and the CLICK clustering software allowed a low-redundancy sequencing—only 7% of the 13 580 clones were sequenced. The full experimental details and results will be reported elsewhere. The DEFOG project is joint work with the groups of H.Lehrach (MPI Berlin) and D.Lancet (Weizmann).

The human sense of smell can detect millions of different odorants—volatile ligands with diverse chemical structures. Olfaction occurs when odorants bind with receptors of olfactory sensory neurons in the nose. The genes encoding odorant receptors form the largest family in the vertebrate genome—roughly 900 OR sequences were found in the first draft of the human genome (Glusman *et al.*, 2001). OR genes have 1Kbp intronless open reading frames, and code for seven-transmembrane domain proteins (Buck and Axel, 1991). They have several highly conserved regions, for example, in transmembrane (TM) segments 2 and 7. By contrast, TM segments 3, 4 and 5 show a high degree of variability—a crucial feature for recognizing a huge variety of odorants (Pilpel and Lancet, 1999). The rest of this section describes the execution of HYDEN and its performance, as part of the aforementioned experiment on the human OR subgenome.

Our experiment began with an initial collection of 127 OR genes, whose full DNA 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. As explained earlier, the first phase of HYDEN constructs an alignment for each k-long substring in the input strings and computes its entropy score. Figure 2 shows the average entropy score for $k = 26$ at each position along the genes. The average entropy at position p is calculated from the entropy scores of the 127 alignments constructed for the sub-strings at position p in the genes. The local minima at positions 160 and 850 correspond to the conserved regions in TM2 and TM7, respectively. These positions make excellent sites for designing PCR primers that amplify a large portion of each gene—almost 700bp. The two other local minima, at positions 350 and 700, are more interior and a little less conserved.

In order to design both $5'$ and $3'$ primers, we ran HYDEN separately on the first and last 300bp of each OR gene. Altogether, we designed 13 primers—6 for the $5'$ side and 7 for the $3'$ side, of lengths $k = 26, 27$ and various degeneracies between 4608 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

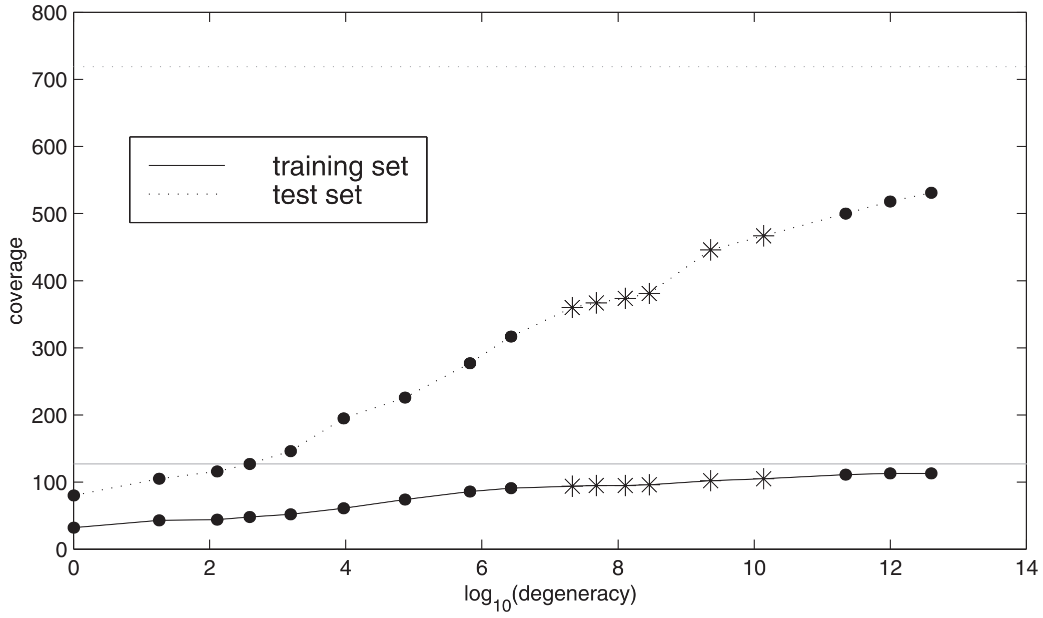

Fig. 3. Training-set and test-set coverage of primer pairs with various degeneracies (allowing up to three mismatches). Primers that were actually used in the DEFOG experiment are marked by asterisks. The gray horizontal lines mark the size of the training and test sets.

reason, are not amplified by the other primers. A typical execution of HYDEN on 300bp segments of the 127 OR genes, with $k = 26$, $d = 20\,000$, and $e = 2$ (and $N_h = 50$, $N_{a'} = 8000$, $N_a = 3000$, $N_g = 100$), takes approximately 10 minutes, distributed evenly among the three phases of the program, on a P4 1.4 GHz PC with 256 MB RDRAM. Except for the special primers, each primer matches 80–90% of the training-set genes with up to two mismatched bases.

From the 13 primers we designed, we selected 20 different pairs, and used them in PCR reactions. The degeneracy of a pair of primers (5′ and 3′) is defined as the product of the degeneracies of both primers. The degeneracy of the pairs we selected ranged between $2.1 \cdot 10^7$ and $1.4 \cdot 10^{10}$. To the best of our knowledge, this is the highest degeneracy ever used successfully in PCR reactions. We also experimented with even higher degeneracies, but their yield was usually very poor (data not shown), 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).

After the publication of the first draft of the human genome, we analysed the performance of the primers on 719 full-length OR sequences (Glusman *et al.*, 2001). 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 3 shows the 3-mismatches coverage of several primer pairs, both for the training set and the test set. 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 have similar degeneracy. The figure also shows the coverage of primers that were not used in the experiment. As expected, primers with higher degeneracy have a larger coverage in both sets. Also apparent is the sharp and steady increase in the test-set coverage as the degeneracy increases—from 10% coverage for non-degenerate primers to 50–65% for the primers we used. 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 in our case—only 0.4% of the clones we sequenced were not OR genes. Second, 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 $1.4 \cdot 10^{10}$.

DISCUSSION

In this paper, we introduced DPD—a combinatorial optimization problem for designing degenerate primers. We developed approximation algorithms for a simplified version of DPD, and implemented HYDEN, an efficient heuristic for the general case. On the theoretical side, one may wish to improve the approximation algorithms in terms of performance (approximation ratio, running time) and generality (input sequences of different lengths, non-binary alphabets, mismatches, gapped alignments). On the practical side, a more realistic primer-gene matching model, which takes into account biological aspects of the PCR procedure (e.g., annealing temperature, position and type of mismatches), could yield primers with greater sensitivity. We are currently extending the HYDEN program to produce *several* primer pairs that together cover all the input sequences.

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. We hope to exploit the utility of degenerate primers on other gene families and other species. In fact, we are currently involved in two projects that use degenerate primers: one for extracting OR genes from a different species and the other for identifying a subfamily of tyrosine kinases from cDNA libraries.

ACKNOWLEDGEMENTS

We thank our friends and collaborators in the DEFOG project without whom this study could not be realized: 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).

R.Shamir was supported in part by grants from the German Israeli Fund (G.I.F.) and from the Israel Science Foundation founded by Israeli Academy of Sciences and Humanities.

REFERENCES

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

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

Doi,K. and Imai,H. (1997) Greedy algorithms for finding a small set of primers satisfying cover length resolution conditions in pcr experiments. *Proceedings of the 8th Workshop on Genome Informatics*. Tokyo, Japan, pp. 43–52.

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

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

Hertz,G.Z. and Stromo,G.D. (1999) Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, **15**, 563–577.

Herwig,R., Schmidt,A.O., 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.

Kwok,S., Chang,S.Y., Sninsky,J.J. and Wang,A. (1994) A guide to the design and use of mismatched and degenerate primers. *PCR Methods and Appl.*, **3**, S39–S47.

Lancet,D. and Ben-Arie,N.R. (1993) Olfactory receptors. *Curr. Biol.*, **3**, 668–674.

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

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

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

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

Sharan,R. and Shamir,R. (2000) CLICK: A clustering algorithm with applications to gene expression analysis. *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB 2000)*. pp. 307–316.