CONTENTS

CHAPTER 3

ALGORITHMIC ISSUES IN DNA BARCODING PROBLEMS (BHASKAR DASGUPTA, MING-YANG KAO, ION MĂNDOIU)

3.1 INTRODUCTION

In the outbreak of an epidemic, possibly as a result of biological warfare, there is an urgent need to identify the pathogen or the family it belongs to as early as possible. Armed with the identity of the pathogen or its family, and prior knowledge of how the pathogen is typically spread, decision makers can efficiently alert the general public and first responders on how best to stave-off the invasion. Recent advances in genomic technologies, including the availability of whole genome sequence for a large number of pathogens and the improved sensitivity of a second generation of microarray-based hybridization platforms, have opened the way for the development of highly reliable genomic-based pathogen detection systems. However, the development of such a detection system appropriate for use by first responders still raises a number of challenging design issues. In addition to portability and cost-effectiveness, widespread use of such systems requires rapid and reliable identification from minute amounts of genetic material of mutated or artificially engineered unknown pathogens. At the same time, these systems should provide comprehensive coverage of known or partially known pathogens, ro-

(String Processing and Application to Biological Sequences, draft). By Mourad Elloumi and Albert Y. Zomaya (eds.)

bustness of the detection algorithms against malicious adversaries and built-in support for easy updates of the set of recognized pathogens.

As a motivation to study a basic version of barcoding problems of interest in this chapter, consider the following scenario. Classical approaches to pathogen detection are based on sequencing and direct microarray hybridization [16, 24]. Although very reliable, sequencing based detection is practically applicable only when the number of candidate pathogens is small, since it requires the ability to isolate a very small number of pathogen-specific DNA or RNA fragments. At the same time, direct microarray hybridization does not scale well with the number of potential pathogens. Reliable detection by this method requires as much as 10-20 arrayed probes per pathogen, each 70 nucleotides long [24], thus limiting the coverage of a single microarray to at most a few thousand pathogens. To overcome some of these difficulties, one employs *rapid* and *robust* computational procedures to compute *barcodes* that produces short signatures and thereby both reduces database size and optimizes cost of designing the hybridization array.

In this chapter, we survey several barcoding problems that have applications as mentioned above as well as in other areas, and survey some key algorithmic techniques used in the existing literatures for these problems. We assume that the reader is familiar with the basic concepts of exact and approximation algorithms (*e.g.*, see [6, 23]), basic computational complexity classes such as P and NP [10, 13, 20] and basic notions of molecular biology such as DNA sequences [12].

3.2 TEST SET PROBLEMS: A GENERAL FRAMEWORK FOR SEVERAL BARCODING PROBLEMS

One of the test set problems was on the classic list of NP-complete problems given by Garey and Johnson [10]; these problems arise naturally in many other applications. One can define a general framework for test set problems in the following manner. We are given an universe of objects, family of subsets ("tests") of the universe and a notion of "distinguishability" of pairs of elements of the universe by a collection of these tests. Our goal is to select a subset of these tests of minimum size that distinguishes every pair of elements of the universe. To be precise, each of these problems is obtained by fixing parameters in the *general* test set problem $\mathrm{TS}^{\Gamma}(k)$ as described below (2^X denotes the *power set* of a set X).

Definition 3.1 (Problem $\mathrm{TS}^{\Gamma}(k)$ with $\Gamma \subseteq 2^{\{0,1,2\}}$ and k being a positive integer)

Instance: $(n, \mathcal{S})$ where $\mathcal{S} \subset 2^{\{0,1,2,\ldots,n-1\}}$.

Terminologies:

- A *k-test* is a union of at most k sets from $\mathcal{S}$.

- For a $\gamma \in \Gamma$ and two distinct elements $x, y \in \{0, 1, 2, \ldots, n-1\}$, a k-test T γ-distinguishes x and y if $|\{x, y\} \cap T| \in \gamma$.

Valid solutions: A collection $\mathcal{T}$ of k-tests such that
$(\forall x, y \in \{0, 1, 2, \ldots, n-1\} \ \ \forall \gamma \in \Gamma) \ \ x \neq y$
$\implies \ \ \exists T \in \mathcal{T}$ such that T γ-distinguishes x and y.

Objective: *minimize* $|\mathcal{T}|$.

This framework captures several "barcoding-type" problems in a few areas in bioinformatics and biological modeling such as:

Minimum Test Collection Problem: This problem has applications in *diagnostic testing* [10]. Here a collection of tests distinguishes two objects if a test from the collection contains exactly one of them. In our above formalism, this is precisely $\mathrm{TS}^{\{1\}}(1)$.

Condition Cover Problem: Karp *et al.* [15] considered a problem of verifying a multi-output feedforward Boolean circuit as a model of biological pathways. This problem can be phrased like the Minimum Test Collection Problem, except that two elements are distinguished by a collection of tests if one tests contains exactly one of them, and another contains both or none of them. Assuming that the allowed perturbations are given as *part of the input*, this problem is identical to $\mathrm{TS}^{\{1\},\{0,2\}}(1)$.

String Barcoding Problem ($\mathrm{SB}^{\Sigma}(k)$): In the "basic" version of this problem corresponding to $k = 1$, first discussed by Rash and Gusfield [21], the universe U consists of sequences (strings) over an alphabet Σ and any string $v \in \Sigma^*$ defines a test T_v consisting of a collection of strings from U in which v appears[1]. Since this chapter is significantly concerned with this basic version, we write the problem definition explicitly for the convenience of the reader. We are given a set $\mathcal{S}$ of sequences over some alphabet Σ. For a *fixed* set of m "distinguisher" sequences $\vec{\mathbf{t}} = (\mathbf{t}_0, \ldots, \mathbf{t}_{m-1})$, the *barcode* $code(s, \vec{\mathbf{t}})$ for each $s \in S$ is defined to be the Boolean vector (c_0, c_1, c_{m-1}) where c_i is 1 if t_i is a substring of s. We say that the set of distinguishers $\vec{\mathbf{t}}$ defines a *valid* barcode if for any two distinct strings $s, s' \in \mathcal{S}$, $code(s, \vec{\mathbf{t}})$ is different from $code(s', \vec{\mathbf{t}})$. Then the basic version $\mathrm{SB}^{\Sigma}(1)$ is defined as follows:

Instance: $\mathcal{S} \subset \Sigma^*$.

Valid solutions: a set of distinguisher sequences $\vec{\mathbf{t}}$ defining a valid barcode.

Objective: *minimize* $|\vec{\mathbf{t}}|$.

[1] Σ^* is the standard notation of denoting the set of all possible strings formed by concatenation of zero or more symbols of Σ.

As an example, let $\Sigma = \{A, C, T, G\}$ and $\mathcal{S} = \{AAC, ACC, GGGG, GTGTGG, TTTT\}$. Then, the set of four distinguishers $\vec{\mathbf{t}} = \{A, CC, TTT, GT\}$ defines the set of valid barcodes for the input sequences in $\mathcal{S}$ as shown below.

	A	CC	TTT	GT
AAC	1	0	0	0
ACC	1	1	0	0
GGGG	0	0	0	0
GTGTGG	0	0	0	1
TTTT	0	0	1	0

The name "string barcoding" derives from the fact that the Boolean vector indicating the occurrence (as a substring) of the tests from an arbitrary collection of tests in a given input sequence is referred to as the "barcode" of the given sequence with respect to this collection of tests. Motivations for investigating these problems come from several sources such as:

- Database compression and fast database search for DNA sequences.
- DNA microarray designs for efficient virus identification in which the immobilized DNA sequences at an array element are from a set of barcodes.

In general, for $k > 1$ a test can be defined by a set T of at most k strings and $u \in U$ passes test T is one of strings in T is a substring of u; such tests may be feasible in practice as the one-string tests.

Minimum Cost Probe Set Problem with a Threshold r ($\mathrm{MCP}^{\Sigma}(r)$):
This problem is very similar to String Barcoding and was considered first by Borneman *et al.* [3]. Denote by $oc(x, y)$ the number of occurrences of x in y as a substring, For a fixed set of m distinguisher sequences $\vec{\mathbf{t}} = (t_0, t_1, \ldots, t_{m-1})$, an r-barcode $code(s, \vec{\mathbf{t}})$ for any sequence s is defined to be the vector $(c_0, c_1, \ldots, c_{m-1})$ where $c_i = \min\{r, oc(t_i, s)\}$. Given a set $\mathcal{S}$ of sequences over some alphabet Σ, $\vec{\mathbf{t}}$ defines a *valid* r-barcode if for any two distinct strings $s, s' \in \mathcal{S}$, $code(s, \vec{\mathbf{t}})$ is different from $code(s', \vec{\mathbf{t}})$. $\mathrm{MCP}^{\Sigma}(r)$ is now defined as follows:

Instance: $(r, \mathcal{S}, \mathcal{P})$ where $\mathcal{S}, \mathcal{P} \subset \Sigma^*$.

Valid solutions: a set of distinguisher sequences $\vec{\mathbf{t}} \subseteq \mathcal{P}$ defining a valid r-barcode.

Objective: *minimize* $|\vec{\mathbf{t}}|$.

This problem was used in [3] for minimizing the number of oligonucleotide probes needed for analyzing populations of ribosomal RNA gene (rDNA) clones by hybridization experiments on DNA microarrays; the probes are selected from a prespecified set $\mathcal{P}$. However, it can also be used in the context of other string barcoding approaches where the barcodes are *integer-valued* as opposed to being Boolean.

3.3 A SYNOPSIS OF BIOLOGICAL APPLICATIONS OF BARCODING

Applications of barcoding techniques range from rapid pathogen identification in epidemic outbreaks to point-of-care medical diagnosis to monitoring of microbial communities in environmental studies (*e.g.*, see [3, 21]). For example, genomic-based identification of microorganisms such as viruses or bacteria is performed by spotting or synthesizing on a microarray the Watson-Crick complements of the distinguisher strings, and then hybridizing to the array the fluorescently labeled DNA extracted from the unknown microorganism. Under the assumption of perfect hybridization stringency, the hybridization pattern can be viewed as a string of k zeros and ones, referred to as the *barcode* of the microorganism. By construction, the barcodes corresponding to the n microorganisms are distinct, and thus the barcode uniquely identifies any one of them. To improve identification robustness, one may also require *redundant distinguishability* (*i.e.*, at least m different distinguishers for every pair of microorganisms, where $m > 1$ is some fixed constant) and impose a lower bound on the edit distance between any pair of selected distinguishers [21].

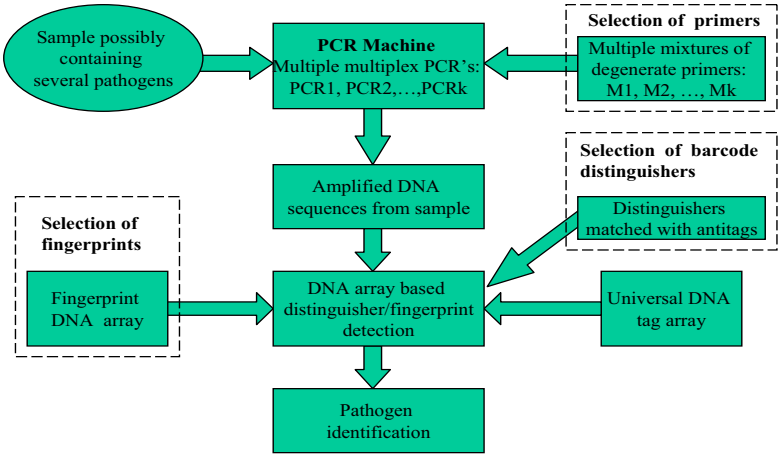

Figure 3.1 *A hypothetical architecture of FRPDS.*

A hypothetical system implementing a high level architecture that meets the design criteria for a *First Responder Pathogen Detection System* (FRPDS) using string barcoding is shown schematically in Figure 3.1. Such a hypothetical system includes the following three major components:

(1) A component that provides *rapid amplification* of the collected genetic material, *e.g.* degenerate oligonucleotide primer based multiplex PCR.

(2) A pathogen fingerprinting and/or barcoding component (say, built around universal DNA tag arrays).

(3) *Rapid* and *robust* computational procedures to compute *barcodes* that produces short signatures and thereby both reduces database size and optimizes cost of designing the hybridization array.

3.4 SURVEY OF ALGORITHMIC TECHNIQUES ON BARCODING

In this section, we survey several algorithmic methods used to solve the barcoding problems. We will then discuss in more details in the next two sections the set-covering and information content algorithmic approach.

3.4.1 Integer Programming

In [21], Rash and Gusfield discussed some experimental results for $\mathrm{SB}^{\Sigma}(1)$ but left open the exact complexity and approximability of this problem. Their algorithmic approach is based on writing the problem as an integer program and then solving it directly. Unfortunately, the run-time of this approach does not scale well with the number of microorganisms and the length of the genomic sequences; *e.g.*, the largest instance sizes reported in [21] have a total genomic sequence length of around $100,000$ bases. We will not discuss the integer programming formulation in more details since we will discuss heuristics based on set-covering methods in more details subsequently and an integer programming formulation for set-covering problem is well-known (*e.g.*, see [23]).

3.4.2 Lagrangian Relaxation and Simulated Annealing

Borneman *et al.* [3] noted that the $\mathrm{MCP}^{\Sigma}(r)$ problem was NP-complete *assuming that the lengths of the sequences in the prespecified set were unrestricted*, and discussed some experimental results for a few heuristics that they implemented. Their algorithmic approach is based on a Lagrangian relaxation of the integer programming formulation of set cover and simulated annealing approach.

3.4.3 Provably Asymptotically Optimal Results

In [2] Berman, DasGupta and Kao were able to provide tight theoretical worst-case approximability bounds for almost all of these problems. A summary of the results in [2] is as follows (ℓ is the maximum length of any sequence in $\mathcal{S}$, L is the total length of all sequences in $\mathcal{S}$, and ε and δ are constants):

- $\mathrm{TS}^{\{1\}}(1)$ can be approximated to within a ratio of $1 + \ln n$ in $O(n^2|\mathcal{S}|)$ time and cannot be approximated to within a ratio of $(1 - \varepsilon)\ln n$ assuming NP$\neq$DTIME$(n^{\log\log n})$.

- $\mathrm{TS}^{\{1\},\{0,2\}}(1)$ can be approximated to within a ratio of $1+\ln 2+\ln n$ in $O(n^2|\mathcal{S}|)$ time and cannot be approximated to within a ratio of $(1-\varepsilon)\ln n$ assuming NP$\neq$DTIME$(n^{\log\log n})$.
- $\mathrm{SB}^{\Sigma}(1)$ can be approximated to within a ratio of $1+\ln n$ in $O(n^3\ell^2)$ time and cannot be approximated to within a ratio of $(1-\varepsilon)\ln n$ assuming NP$\neq$DTIME$(n^{\log\log n})$.
- $\mathrm{MCP}^{\Sigma}(r)$ can be approximated to within a ratio of $[1+o(1)]\ln n$ in $O(n^2|\mathcal{P}|+L\mathcal{P}|)$ time and cannot be approximated to within a ratio of $(1-\varepsilon)\ln n$ assuming NP$\neq$DTIME$(n^{\log\log n})$.
- $\mathrm{TS}^{\{1\}}(n^{\delta})$ cannot be approximated to within a ratio of n^{ε} assuming NP$\neq$co-RP for any $0<\varepsilon<\delta<1$.
- $\mathrm{SB}^{\Sigma}(n^{\delta})$ cannot be approximated to within a ratio of n^{ε} assuming NP$\neq$co-RP for any $0<\varepsilon<\delta<\frac{1}{2}$.

The provably optimal algorithmic approach in [2] uses an entropy-based algorithmic approach that they term as the "information content" approach. Informally, this is a greedy technique based on information content of a partial solution; the notion of information content is directly related to the Shannon information complexity [1, 22]. The greedy approach seeks to select an augmenting step for a partial solutions that maximizes the *new* information content of the augmented partial solution as compared to the partial solution. A key non-trivial step for applicability of this technique is to define a suitable easy-to-compute measure of the information content of a partial solution such that the monotonicity of this measure is ensured with respect to any subset of an optimal solution. The next section defines the approach more precisely.

3.5 INFORMATION CONTENT APPROACH

In this section we discuss the information content approach for $\mathrm{TS}^{\{1\}}$ as designed in [2] running in time $O(n^2|\mathcal{S}|)$ time with an approximation ratio of $1+\ln n$. Notice that the upper bound almost matches the lower bound stated in Section 3.4.3 for $\mathrm{SB}^{\{0,1\}}$, a special case of $\mathrm{TS}^{\{1\}}$.

For simplicity, we illustrate the approach for the problem $\mathrm{TS}^{\{1\}}$. In the definition below and throughout the rest of this section we use $\mathcal{T}+T$ to denote $\mathcal{T}\cup\{T\}$.

Definition 3.1 *A set of tests $\mathcal{T}\subseteq\mathcal{S}$ defines the following:*

- *an equivalence relation $\overset{\mathcal{T}}{\equiv}$ on $\{0,1,2,\ldots,n-1\}$ given by $i\overset{\mathcal{T}}{\equiv}j$ if and only if $\forall T\in\mathcal{T}(i\in T\equiv j\in T)$,*
- *a set of permutations $\Pi_{\mathcal{T}}=\{\pi\in(\text{permutations of }\{0,1,2,\ldots,n-1\}):\ \forall i\in[0,n-1]\ i\overset{\mathcal{T}}{\equiv}\pi(i)\}$,*

- *entropy* $H_{\mathcal{T}} = \log_2 |\Pi_{\mathcal{T}}|$.
- *information content of a* $T \in \mathcal{S}$ *with respect to* $\mathcal{T}$, $IC(T, \mathcal{T}) = H_{\mathcal{T}} - H_{\mathcal{T}+T} = \log_2 \frac{|\Pi_{\mathcal{T}}|}{|\Pi_{\mathcal{T}+T}|}$.

As an example, consider $\mathcal{T} = \{\{1,2,3,4\},\{1,5,6\}\}$ with $n = 8$. Then, the equivalence classes of $\overset{\mathcal{T}}{\equiv}$ are $\{1\},\{2,3,4\},\{5,6\},\{7,8\}$ and $H_{\mathcal{T}} = \log_2((3!)(2!)(2!)) \approx 4.585$. The above definition of entropy is somewhat similar (but not the same) to the one suggested in [18]. Suppose that the equivalence relation $\overset{\mathcal{T}}{\equiv}$ on $\{0,1,2,\ldots,n-1\}$ produces q equivalence classes of size $s_1, s_2, \ldots, s_q$. Then, the entropy suggested in [18] is $\frac{1}{n}\log_2(\Pi_{i=1}^{q} s_i^{s_i})$ whereas our entropy $H_{\mathcal{T}}$ is $\log_2(\Pi_{i=1}^{q} s_i!)$.

The *information content heuristic* (ICH for short) is the following simple greedy heuristic:

```
T = ∅
while H_T ≠ 0 do
        select a T ∈ S − T that maximizes IC(T, T)
        T = T + T
```

The correctness of ICH follows from the fact that $H_{\mathcal{T}} = 0$ implies the equivalence classes of $\overset{\mathcal{T}}{\equiv}$ are n singleton sets $\{0\},\{1\},\ldots,\{n-1\}$ and the fact that if $H_{\mathcal{T}} \neq 0$ then there exists a $T \in \mathcal{S} \setminus \mathcal{T}$ with $IC(T, \mathcal{T}) > 0$ (otherwise the problem instance has no feasible solution).

To implement ICH, one iteratively maintains the equivalence classes of $\overset{\mathcal{T}}{\equiv}$ as sorted lists. We also precompute and store $\log_2(i!)$ for each $i \in [1,n]$. Given a specific $T \in \mathcal{S} - \mathcal{T}$, it is easy to compute in $O(n)$ time the equivalence classes of $\overset{\mathcal{T}+T}{\equiv}$ from the equivalence classes of $\overset{\mathcal{T}}{\equiv}$ since an equivalence class E of $\overset{\mathcal{T}}{\equiv}$ is either an equivalence class of $\overset{\mathcal{T}+T}{\equiv}$ or it is partitioned into two equivalence classes $E_1 = E \cap T$ and $E_2 = E - E_1$ of $\overset{\mathcal{T}+T}{\equiv}$; the first case contributes nothing to $IC(T, \mathcal{T})$ while the second case adds $\log_2 \binom{|E|}{|E_1|}$ to $IC(T, \mathcal{T})$.

The performance guarantee of the above approach is given by the following theorem proved in [2] using a very careful amortized analysis.

Theorem 3.1 [2] *The above approach yields:*

- *for* $TS^{\{\{1\}\}}$ *an approximation ratio of* $1 + \ln n$;
- *for* $TS^{\{\{1\},]\{0,2\}\}}$ *an approximation ratio of* $1 + \ln 2 + \ln n$;
- *for* $MCP^{\Sigma}(r)$ *an approximation ratio of* $1 + \ln n + \ln \log_2(r'+1)$, *where* $r' = \min\{r, n\}$.

3.6 SET COVERING APPROACH

Methods based on this approach enable distinguisher selection based on *whole genomic sequences* of hundreds of microorganisms of up to bacterial size on a well-equipped workstation, and can be easily parallelized to further extend the applicability range to thousands of bacterial size genomes. Whole-genome based selection is beneficial in at least two significant ways. First, it simplifies assay design since the DNA of the unknown pathogen can be amplified using inexpensive general-purpose whole-genome amplification methods such as specialized forms of degenerate primer multiplex PCR [4] or multiple displacement amplification [9]. Second, whole-genome based selection results in a reduced number of distinguishers, often very close to the information theoretic lower bound of $\lceil \log_2 n \rceil$.

Set covering approaches are based on a simple greedy selection strategy – in every iteration we pick a substring that distinguishes the largest number of not-yet-distinguished pairs of genomic sequences. This selection strategy is an embodiment of the greedy setcover algorithm (*e.g.*, see [23]) for a problem instance with $O(n^2)$ elements corresponding to the pairs of sequences. Hence, by a classical result of [5, 14, 17], the algorithm guarantees an approximation factor of $2 \ln n$ for the barcoding problem. Experimental results provided in [7, 8] show that our setcover greedy algorithm produces solutions of virtually identical quality to those obtained by the information content heuristic.

The setcover greedy algorithm is extremely versatile, and can be easily extended to handle redundancy and minimum edit distance constraints, as well as other biochemical constraints on individual distinguisher sequences. Furthermore, the greedy setcover algorithm can also take into account genomic sequence uncertainties expressed in the form of degenerate bases. Although degenerate bases are ubiquitous in genomic databases, previous works have not recognized the need to properly handle them.

3.6.1 Set Covering Implementation in More Details

In this section for simplicity we present the implementation of the setcover greedy algorithm as provided in [7, 8] in the context of the basic version of the string barcoding problem only. Implementation modifications needed to handle the robust barcoding problem in its full generality are available in [8].

The implementation of the setcover greedy algorithm has two main phases: a *candidate generation phase* and a *candidate selection phase*.

In the candidate generation phase a representative set of candidate distinguishers is generated from the given genomic sequences. Essentially they use an incremental algorithm for quickly generating a representative set of candidate distinguishers and collecting all their occurrences in the given genomic sequences. For each generated candidate, we also compute the list of sequences with which the candidate has perfect matches; this information is needed in the candidate selection phase. To reduce the number of candidates, we avoid

generating any substring that appears in all genomic sequences, which typically eliminates very short candidates. For each genomic sequence, we also make sure to generate only one of the substrings that appear exclusively in that sequence; this optimization eliminates from consideration most candidate distinguishers above a certain length. Unlike the suffix tree method proposed by Rash and Gusfield [21], this approach may generate multiple candidates that appear in the same set of k genomic sequences (for $1 < k < n$). However, the penalty of having to evaluate redundant candidates in the candidate selection phase is offset in practice by the faster candidate generation time. Efficient implementation of the above candidate elimination rules is achieved by generating candidates in increasing order of length and using exact match positions for candidates of length $l-1$ when generating candidates of length l. For each position p in the input genomic sequences, we also maintain a flag to indicate whether or not the algorithm should evaluate candidate substrings starting at p. The possible values for the flag are TRUE (the substring of current length starting at p is a possible candidate), FALSE (we have already saved the substring of current length starting at p as a candidate), or DONE (all candidates containing as prefix the substring of current length starting at p are redundant, i.e., the position can be skipped for all remaining candidate lengths). Initially all flags are set to TRUE. The FALSE flags are reset to TRUE whenever we increment the candidate length, however, we never reset DONE flags.

For every candidate length l, candidate evaluation proceeds sequentially over all positions of the genomic sequences. Whenever we reach a position p whose flag is set to TRUE, we use the list of matches for the substring of length $l-1$ starting at p (or a linear time string matching algorithm if l is the minimum candidate length) to determine the list of matches for the substring of length l starting at p, and set the flag to FALSE for all positions where these matches occur. If the substring of length l starting at p has matches only within the source sequence, and we have already generated a "unique" candidate for this sequence, we discard the candidate and set the flag of p to DONE.

A further speed-up technique is to generate candidate distinguishers from a strict subset of the input sequences. Although this speed-up can potentially affect solution quality, experimental results show that the solution quality loss for whole-genome barcoding is minimal, even when we generate candidates based on a single input sequence, which corresponds to pre-assigning a barcode of all 1's to this sequence.

After the set of candidates is generated we select the final set of distinguishers in the greedy phase of the algorithm (Figure 3.2). We start with an empty set of distinguishers D. While there are pairs of sequences that are not yet distinguished by D, we loop over all candidates and compute for each candidate c the number $\Delta(c, D)$ of pairs of sequences that are distinguished by c but not by D, then add the candidate c with largest Δ value to D.

Input: Set C of candidate distinguishers
Output: Set D of selected distinguishers

$D \leftarrow \emptyset$; For every $c \in C$, $\Delta_{old}(c) \leftarrow \infty$
Repeat
  $\Delta^* \leftarrow 0$
  For every $c \in C$ with $\Delta_{old}(c) > \Delta^*$ do // Since $\Delta(c, D) \leq \Delta_{old}(c)$, c can be ignored if $\Delta_{old}(c) \leq \Delta^*$
    $\Delta_{old}(c) \leftarrow \Delta(c, D)$
    If $\Delta(c, D) > \Delta^*$ then $\Delta^* \leftarrow \Delta(c, D)$; $c^* \leftarrow c$
  If $\Delta^* > 0$ then $D \leftarrow D \cup \{c^*\}$
While $\Delta^* > 0$
Return D

Figure 3.2 The greedy candidate selection algorithm

Two sequences s and s' are distinguished by a candidate c if and only if exactly one of s and s' appears in the list P_c of perfect matches of c, which is available from the candidate generation phase. A simple method for computing Δ values is to maintain an $n \times n$ symmetric matrix indicating which of the pairs of sequences are already distinguished, and then to probe the $|P_c| \cdot (n - |P_c|)$ entries in this matrix corresponding to pairs (s, s') with $s \in P_c$ and $s' \notin P_c$ when computing $\Delta(c, D)$. A more efficient method is based on maintaining the partition defined on the set of sequences by D. If the partition defined by D consists of sets $S_1, \ldots, S_k$, then we can compute $\Delta(c, D)$ in $O(k + |P_c|) = O(n)$ time using the observation that

$$\Delta(c, D) = \sum_{i=1}^{k} |S_i \cap P_c| \cdot |S_i \setminus P_c| \tag{3.1}$$

In addition to the fast partition based computation, the implementation of the greedy selection phase uses a lazy strategy for updating the Δ values, based on the observation that they are monotonically non-increasing during the algorithm (see Figure 3.2). Thus, the efficient implementation of the greedy selection phase of the algorithm combines a partition based method for computing the coverage gain of candidate distinguishers (this method was first proposed in the context of the information content heuristic in [2]) with a "lazy" strategy for updating coverage gains.

3.7 EXPERIMENTAL RESULTS AND SOFTWARE AVAILABILITY

The authors in [7, 8] performed experiments on both randomly generated instances and whole microbial genomes extracted from the NCBI databases. Random testcases were generated from the uniform distribution induced by

assigning equal probabilities to each of the four nucleotide; these testcases do not contain any nucleotides with degeneracy greater than 1. The NCBI testcase represents a selection of 29 complete microbial sequences, varying in length between 490,000 and 4,750,000 bases (over 76 million bases in total). All experiments were run on a PowerEdge 2600 Linux server with 4 Gb of RAM and dual 2.8 GHz Intel Xeon CPUs – only one of which is used by the sequential algorithms.

3.7.1 Randomly Generated Instances

As described in Section 3.6.1, there are two main phases in the algorithm: candidate distinguisher generation, and greedy candidate selection. Results were reported about the average candidate selection CPU time for n random sequences of length 10,000 and redundancy 1, averaged over 10 instances of each size. Combining the two speed-up techniques for this phase (partition based coverage gain computation and lazy update of candidate gains) results in over two orders of magnitude reductions in runtime.

A further speed-up technique is to generate candidate distinguishers from a select subset of the input sequences. Although this speed-up can potentially affect solution quality, the results showed that on large instances the solution quality loss is minimal even when we generate candidates based on a single input sequence, this corresponds to pre-assigning a barcode of all 1's to this sequence. The technique reduces significantly both the memory requirement (which is proportional to the number of candidates and the number of times they match input sequences) and the runtime required for candidate generation and greedy selection.

The quality of the solution in the simulations were as follows. The number of distinguishers returned by the setcover greedy algorithm were reported for redundancy varying between 1 and 20 on between 10 and 1,000 random sequences of length 10,000. These results were compared with the results obtained by the information content heuristic results of [2], as well as the information theoretic lower bound of $\lceil \log_2 n \rceil$ for the case when the redundancy requirement is 1. The number of distinguishers returned by the setcover greedy algorithm was virtually identical to that returned by the information content heuristic, despite the latter one having a better approximation guarantee. Furthermore, the results for redundancy one were within 50% of the information theoretic lower bound for the range of instance sizes considered in this experiment. The gap between the solutions returned by the algorithms and the lower bound does increase with the number of sequences; however it is not clear how much of this increase is caused by degrading algorithm solution quality, and how much by degrading lower bound quality.

3.7.2 Real Data

The algorithm was run on a set of 29 complete microbial genomic sequences extracted from NCBI databases [19]. Sequence lengths in the set vary between 490 Kbases and 4.75 Mbases, with an average length of 2.6 Mbases (over 76 Mbases total). In these experiments we varied the redundancy requirement from 1 to 20. To see the effect of length and edit distance requirements on the number of distinguishers, for each redundancy requirement they computed both an unconstrained solution, and a solution in which distinguishers must have length between 15 and 40, and there should be a minimum edit distance of 6 between every two selected distinguishers (these values are similar to those used in [21]. In all experiments, they generated candidates based only on the shortest sequence of 490 Kbases.

Naturally, meeting higher redundancy constraints requires more distinguishers to be selected. Additional length and edit distance constraints further increase the number of distinguishers, but the latter is still within reasonable limits. The length constraints reduce the number of candidates (from 1,775,471 to 122,478), which, for low redundancy values has the effect of reducing greedy selection time. However, for high redundancy requirements the reduction in number of candidates is offset by the increase in solution size, and greedy selection becomes more time consuming with length and edit distance than without (selection time grows roughly linearly with solution size).

3.7.3 Software Availability

The implementation of the set-covering approach, which was named DNA-BAR, can be used online through the web interface provided at `http://dna.engr.uconn.edu/~software/DNA-BAR/`. The open source C code, released under the GNU General Public License, is also available at the above address.

3.8 CONCLUDING REMARKS

In many practical pathogen identification applications collected biological samples may contain the DNA of multiple pathogens. This issue is considered to be particularly significant in medical diagnosis applications (*e.g.*, see [11] for studies in detecting more than one HPV (human papilomavirus) genotype with varying rate of multiple HPV infections carried by the same HPV carrier). A significant future research direction could be to develop extensions of the barcoding technique that can reliably detect multiple pathogens for a given bound on the number of pathogens present.

Acknowledgments

Bhaskar DasGupta was supported in part by NSF grants IIS-0346973, IIS-0612044 and DBI-0543365. Ion Măndoiu was supported in part by NSF grant DBI-0543365. The authors would also like to thank all their collaborators in these research topics.

References

1. Y. S. Abu-Mostafa (editor). *Complexity in Information Theory*, Springer Verlag, 1986.
2. P. Berman, B. DasGupta and M.-Y. Kao. *Tight Approximability Results for Test Set Problems in Bioinformatics*, Journal of Computer & System Sciences, 71 (2), 145-162, 2005.
3. J. Borneman, M. Chrobak, G. D. Vedova, A. Figueora and T. Jiang. *Probe Selection Algorithms with Applications in the Analysis of Microbial Communities*, Bioinformatics, 1, 1-9, 2001.
4. V.G. Cheung and S.F. Nelson. *Whole genome amplification using a degenerate oligonucleotide primer allows hundreds of genotypes to be performed on less than one nanogram of genomic dna*, PNAS, 93, 14676-14679, 1996.
5. V. Chvátal. *A greedy heuristic for the set covering problem*, Mathematics of Operations Research, 4, 233-235, 1979.
6. T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms*, The MIT Press, 2001.
7. B. DasGupta, K. Konwar, I. Mandoiu and A. Shvartsman. *DNA-BAR: Distinguisher Selection for DNA Barcoding*, Bioinformatics, 21 (16), 3424-2426, 2005.
8. B. DasGupta, K. Konwar, I. Mandoiu and A. Shvartsman. *Highly Scalable Algorithms for Robust String Barcoding*, International Journal of Bioinformatics Research & Applications, 1 (2), 145-161, 2005.

9. F.B. Dean, S. Hosono, *et al. Comprehensive human genome amplification using multiple displacement amplification*, PNAS, 99, 5261-5266, 2002.
10. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness,* W. H. Freeman, 1979.
11. B. Gharizadeh, M. Käller, P. Nyrén, A. Andersson, M. Uhlén, J. Lundeberg, and A. Ahmadian. *Viral and microbial genotyping by a combination of multiplex competitive hybridization and specific extension followed by hybridization to generic tag arrays*, Nucleic Acids Research, 31 (22), 2003.
12. D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge Univ Press, 1997.
13. D. Hochbaum. *Approximation Algorithms for NP-hard Problems*, PWS publishers, 1996.
14. D.S. Johnson. *Approximation algorithms for combinatorial problems* Journal of Computer & System Sciences, 9, 256-278, 1974.
15. R. M. Karp, R. Stoughton and K. Y. Yeung. *Algorithms for Choosing Differential Gene Expression Experiments*, Proc. Third Annual International Conference On Computational Molecular Biology, 208-217, 1999.
16. T. G. Ksiazek, D. Erdman, *et al. A novel coronavirus associated with severe acute respiratory syndrome*, N. Engl. J. Med. 348, 1953-1966, 2003.
17. L. Lovász. *On the ratio of optimal integral and fractional covers*, Discrete Mathematics, 13, 383-390, 1975.
18. B. M. E. Moret and H. D. Shapiro. *On minimizing a set of tests*, SIAM Journal on Scientific and Statistical Computing, 6, 983-1003, 1985.
19. NCBI Completed Microbial Genomes, `http://www.ncbi.nlm.nih.gov/genomes/microbes/complete.html`, Oct. 2004.
20. C. H. Papadimitriou. *Computational Complexity*, Addison-Wesley; reading, MA, 1994.
21. S. Rash and D. Gusfield. *String Barcoding: Uncovering Optimal Virus Signatures*, Sixth Annual International Conference on Computational Biology, 54-261, 2002.
22. C. E. Shannon. *Mathematical Theory of Communication*, Bell Systems Technical Journal, 379-423, 1948.
23. V. Vazirani. *Approximation Algorithms*, Springer-Verlag, July 2001.
24. D. Wang, L. Coscoy, *et al. Microarray-based detection and genotyping of viral pathogens*, PNAS, 99, 15687-15692, 2002.