Highly Scalable Algorithms for Robust String Barcoding*

Bhaskar DasGupta[†]
Kishori M. Konwar[‡§]
Ion I. Măndoiu[‡¶]
Alex A. Shvartsman[‡§]

[†]Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607-7053. E-mail: dasgupta@cs.uic.edu. Supported in part by NSF grants CCR-0206795, CCR-0208749 and NSF CAREER grant IIS-0346973.
[‡]Computer Science and Engineering Department, University of Connecticut, 371 Fairfield Rd., Unit 2155, Storrs, CT 06269-2155. E-mail: {kishori,ion,aas}@cse.uconn.edu.
[§]Supported in part by NSF ITR grant 0121277.
[¶]Corresponding author. Supported in part by a "Large Grant" from the University of Connecticut's Research Foundation.
[*]Authors are listed in alphabetical order.

Abstract: String barcoding is a recently introduced technique for genomic-based identification of microorganisms. In this paper we describe the engineering of highly scalable algorithms for robust string barcoding. Our methods enable distinguisher selection based on *whole genomic sequences* of hundreds of microorganisms of up to bacterial size on a well-equipped workstation. Experimental results on both randomly generated and NCBI genomic data show that whole-genome based selection results in a number of distinguishers nearly matching the information theoretic lower bounds for the problem.

Keywords: String barcoding, setcover problem, greedy algorithm

Reference to this paper should be made as follows: DasGupta, B., Konwar, K.M., Măndoiu, I.I., and Shvartsman, A.A. (xxxx) 'Highly Scalable Algorithms for Robust String Barcoding', *Int. J. of Bioinformatics Research and Applications*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes:
Bhaskar DasGupta received his PhD from University of Minnesota. Subsequently he held post-doctoral positions at the DIMACS institute of Rutgers University, and at University of Waterloo and McMaster University in Canada before joining the Camden campus of Rutgers University as a faculty. Currently, he is an assistant professor in the University of Illinois at Chicago. His main research interests include designing combinatorial approximation algorithms for computationally challenging problems in bioinformatics and several other areas. His research has been supported by several NSF research grants. He was the recipient of the NSF CAREER award in 2004.

Kishori M. Konwar received the M.Sc. degree in physics from the Indian Institute of Technology, Kanpur, India, in 1998 and the M.Tech. degree in computer science from the Indian Statistical Institute, Calcutta, India, in 2000. Presently, he is pursuing the Ph.D. degree at the University of Connecticut, Storrs.

Ion I. Măndoiu received the M.S. degree from Bucharest University in 1992 and the Ph.D. degree from Georgia Institute of Technology in 2000, both in Computer Science. He is now an Assistant Professor with the Computer Science and Engineering Department at the University of Connecticut. His research focuses on the design and analysis of exact and approximation algorithms for NP-hard optimization problems, particularly in the areas of bioinformatics and computational biology, VLSI computer aided design, and ad-hoc wireless notworks.

Alexander Allister Shvartsman is on the faculty of Computer Science and Engineering at the University of Connecticut and he is a research associate at the Laboratory for Computer Science of the Massachusetts Institute of Technology. Previously he was a Member of Technical Staff at Bell Labs from 1981 to 1982, and later he had led the development of distributed systems in the areas of manufacturing automation, resource management and interactive multimedia at Digital Equipment Corp. from 1983 to 1994 and Logica, Inc. from 1994 to 1995. His primary research interests are in the principles and practices of dependable distributed computing. He graduated secondary school in 1972 in Chişinău, Moldova. He received a B.S. from Stevens Institute of Technology in 1979, an M.S. from Cornell University in 1981, and a Ph.D. from Brown University in 1992, all in Computer Science.

1 Introduction

String barcoding is a recently introduced technique for genomic-based identification of microorganisms such as viruses or bacteria. The basic barcoding problem [15] is formulated as follows: given the genomic sequences $g_1, \ldots g_n$ of n microorganisms, find a minimum number of strings $t_1, \ldots, t_k$ *distinguishing* these genomic sequences, i.e., having the property that, for every $g_i \neq g_j$, there exists a string t_l which is a substring of g_i or g_j, but not of both. A closely related formulation was independently proposed in [3], where it is assumed that it is possible to detect not just the presence or absence of a distinguisher t_i, but also the number of repetitions of t_i as a substring, up to a threshold of $R > 0$. The formulation in [15], which we adopt in this paper, corresponds to $R = 1$.

Identification is performed by spotting or synthesizing on a microarray the Watson-Crick complements of the distinguisher strings $t_1, \ldots, t_k$, 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 [15].

The algorithms previously proposed for string barcoding are based on integer programming [15], and on Lagrangian relaxation and simulated annealing [3]. Unfortunately, the run-time of these algorithms does not scale well with the number of microorganisms and the length of the genomic sequences, e.g., the largest instance sizes reported in [15] have a total genomic sequence length of around 100,000 bases.

In this paper we describe the engineering of highly scalable algorithms for robust string barcoding. Our methods 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 [5] or multiple displacement amplification [8]. Second, whole-genome based selection results in a reduced number of distinguishers, often very close to the information theoretic lower bound of $\lceil \log_2 n \rceil$.

Our algorithms 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 (see, e.g., [17]) for a problem instance with $O(n^2)$ elements corresponding to the pairs of sequences. Hence, by a classical result of [6, 11, 13], our algorithm guarantees an approximation factor of $2 \ln n$ for the barcoding problem. Very recently, Berman et al. [1] have shown that no approximation algorithm can guarantee a factor of $(1 - \epsilon) \ln n$ unless $NP = DTIME(n^{\log \log n})$, and also proposed an information content greedy heuristic achieving an approximation factor of $1 + \ln n$. Experimental results given in Section 5 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, unlike the information content heuristic of [1], 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. For example, experiments in [15] have implicitly treated degenerate bases in the input genomic sequences as distinct nucleotides; under this approach a substring of degenerate nucleotides such as NNNNN, might be erroneously selected as a distinguisher although it encodes for any possible substring of length 5.

To achieve high scalability, our implementation relies on several techniques. First, we use an incremental algorithm for quickly generating a representative set of candidate distinguishers and collecting all their occurrences in the given genomic sequences. 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 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 [15], our 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. Finally, the efficient implementation of the greedy selection phase of 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 [1]) with a "lazy" strategy for updating coverage gains.

The rest of the paper is organized as follows. In Section 2 we give formal problem formulations and review previous work. In Section 3 we describe the efficient implementation of the setcover greedy algorithm for the basic string barcoding problem. In Section 4 we discuss the modifications required in the implementation for handling degenerate bases in input genomic sequences, redundancy and edit distance constraints, as well as biochemical constraints such as constraints on melting temperature and GC-content. In Section 5 we give the results of a comprehensive experimental study comparing, on both randomly generated and genomic data, our setcover greedy algorithms with other scalable methods including the information content heuristic and a recent set multicover randomized rounding approximation algorithm. We conclude in Section 6 with directions for further research.

2 Preliminaries and Problem Formulation

Let $\Sigma = \{a, c, g, t\}$ be the DNA alphabet, and Σ^* be the set of string over Σ. A *degenerate base* is a non-empty subset of Σ. We identify degenerate bases of cardinality 1 with the respective non-degenerate bases. Given a DNA string $x = x_1 \dots x_k \in \Sigma^*$ and a string of degenerate bases $y = y_1 \dots y_n$, $n \geq k$, we say that

- x has a *perfect match* at position i of y iff $y_{i+j-1} = \{x_j\}$ for every $1 \leq j \leq k$,
- x has a *perfect mismatch* at position i of y iff there exists $1 \leq j \leq k$ such that $\{x_j\} \not\subseteq y_{i+j-1}$, and
- x has an *uncertain match* at position i of y iff $\{x_j\} \subseteq y_{i+j-1}$ for every $1 \leq j \leq k$, but $y_{i+j-1} \neq \{x_j\}$ for at least one j.

String $x = x_1 \dots x_k \in \Sigma^*$ *distinguishes* two sequences of degenerate bases y and z iff (a) x has a perfect match at one or more positions of y, and has perfect mismatches at all positions of z, or, symmetrically, (b) x has a perfect match at one or more positions of z, and has perfect mismatches at all positions of y. The robust string barcoding problem with degenerate bases is formulated as follows: *Given sequences of degenerate bases $g_1, \dots g_n$ and redundancy threshold m, find a minimum number of strings $t_1, \dots, t_k \in \Sigma^*$ such that, for every $i \neq j$, there exist m distinct strings t_l distinguishing g_i and g_j.*

It is easy to see that, for $m = 1$, at least $\lceil \log_2 n \rceil$ distinguishers are needed to distinguish any n genomic sequences. However, achieving this lower bound requires distinguishers that have perfect matches in nearly half of the sequences. In practice, additional constraints, such as lower bounds on the length of distinguishers, may result in no string having perfect matches in a large number of sequences, and therefore much more than a logarithmic number of distinguishers. The next theorem, the proof of which we omit due to space constraints, establishes under a simple probabilistic model that there is an abundance of distinguishers perfectly matching at least a constant fraction of the input sequences.

Theorem 1 *Consider a random instance of the string barcoding problem over a fixed alphabet Σ in which there are n strings, each string $s = s_0 s_1 \dots s_{\ell-1}$ is of length exactly ℓ selected independently at random with $Pr[s_i = a] = 1/|\Sigma|$ for any i and any $a \in \Sigma$. Also assume that ℓ is sufficiently large compared to n. Then, for a random string $x \in \Sigma^*$ of length $O(\log \ell)$, the expected number of the input strings which contain x as a substring is pn for some constant $0 < p < 1$.*

Proof. Assume n and ℓ to be sufficiently large for asymptotic results and $\sigma = |\Sigma| > 1$ to be fixed. It suffices to show that for a random string $x \in \Sigma^*$ of length $k = O(\log \ell)$, $Pr[x \text{ is a substring of } s] = p$ for some constant $0 < p < 1$ and s is any one of the input n strings. In [14]Examples 6.4, 6.7, 6.8, 9.3 and 10.11, Odlyzko uses the bounds and generating function described in [10] to give asymptotic bounds on $Pr[x \text{ is a substring of } s]$ when $\sigma = 2$. The result can be generalized to the case of any fixed $\sigma > 2$ as follows. For a fixed $x = x_1 x_2 \ldots x_k$, define the correlation polynomial $C_x(z)$ of x as $C_x(z) = \sum_{j=0}^{k-1} c_x(j) z^j$ where $c_x(0) = 1$ and, for $1 \le j < k$,

$$c_x(j) = \begin{cases} 1 & \text{if } x_1 x_2 \ldots x_{k-j} = x_{j+1} x_{j+2} \ldots x_k \\ 0 & \text{otherwise} \end{cases}$$

Let $f_x(\ell)$ be the number of strings in Σ^* of length ℓ that do not contain x as a substring and $F_x(z) = \sum_{\ell=0}^{\infty} f_x(\ell) z^\ell$ be the generating function for this number. Then, $F_x(z) = \frac{C_x(z)}{z^k + (1-\sigma z) C_x(z)}$. From this, it follows that

$$Pr[x \prec s] = 1 - \mathbf{e}^{-\frac{\ell}{\sigma^k C_x(1/\sigma)} + O(\ell k \sigma^{-\sigma k})} + O(\mathbf{e}^{-\ell/O(1)})$$

for all sufficiently large n, k and ℓ, where $\mathbf{e}$ is the base of natural logarithm. Note that $1 \le C_x(\sigma) < 2$ and for a specific x, $C_x(\sigma)$ can be calculated exactly. Now, setting $k = \Theta(\log_\sigma \ell)$ gives $Pr[x \text{ is a substring of } s] = p$ for some constant $0 < p < 1$. ❑

Previous work. The robust string barcoding problem was introduced (for the case when genomic sequences contain no degenerate bases) by Rash and Gusfield [15]; they provided some experimental results based on integer programming methods, and left open the exact complexity and approximability of this problem. The problem without redundancy constraints was independently considered by Borneman *et al.* [3], who also considered non-binary distinguishability (based on detecting the multiplicity of a distinguisher as a substring) and a slightly more general problem in which the objective is to pick a given number of distinguishers maximizing the number of distinguished pairs. The main motivation for the formulations in [3] comes from minimizing the number of oligonucleotide probes needed for analyzing populations of ribosomal RNA gene (rDNA) clones by hybridization experiments on DNA microarrays. Borneman *et al.* provided computational results using Lagrangian relaxation and simulated annealing techniques, and noted that the problem is NP-hard assuming that the lengths of the sequences in the prespecified set were unrestricted. Very recently, Berman, DasGupta and Kao [1] considered a general framework for test set problems that captured the string barcoding problem and its variations; their main contribution is to establish theoretically matching lower and upper bounds on the worst-case approximation ratio. Cazalis et al. [4] have independently investigated similar greedy distinguisher selection strategies for string barcoding. Unlike our work, the algorithms in [4] consider only a small random subset of the possible distinguishers and also prescribe their length in order to achieve practical running time.

3 Efficient Implementation of the Greedy Setcover Algorithm

In this section we present the implementation of the setcover greedy algorithm in the context of the basic string barcoding problem, i.e., we disregard redundancy constraints and the presence of degenerate bases in the input sequences. Implementation modifications needed to handle the robust barcoding problem in its full generality are discussed in Section 4.

Our 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. 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 [15], our 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

Figure 1 The setcover greedy candidate selection algorithm

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 $\Delta^* > 0$
```

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, the results in Section 5 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 1). 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. Two sequences s and s' are distinguished by a candidate c iff 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

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

In addition to the fast partition based computation, our 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 1).

4 Extended Barcoding Requirements

In this section we describe the modifications needed to the basic implementation given in previous section when handling practical extensions of the barcoding problem.

Degenerate bases. In the presence of degenerate bases in the input genomic sequences, the hybridization of a particular distinguisher may depend on which bases are actually present positions with degeneracy greater than 1. The greedy setcover algorithm takes into account this possibility for uncertain hybridization by only counting a pair (g, g') as distinguished by a candidate c if and only if c has a perfect match with one and only perfect mismatches with the other. For each generated candidate, in addition to the list of sequences that have only perfect matches we also save a list containing all sequences with at least one uncertain match. This allows fast computation of the (typically much longer) list of sequences having only perfect mismatches. To avoid generating candidate distinguishers containing degenerate bases, we set the DONE flag as soon as the corresponding substring extends past a degenerate base. Finally, since the partition of genomic sequences is no longer defined in the presence of uncertain hybridization, formula (1) is no longer applicable and we have to use the $n \times n$ "distinguished so far" matrix for computing Δ values.

Biochemical constraints on individual distinguishers. Since selected distinguishers must hybridize under the same experimental conditions, in practice it is natural to impose a variety of constraints on individual distinguishers, such as minimum and maximum length, GC content, melting temperature, etc. Furthermore, we may want to avoid using as distinguishers strings which appear in other organisms that may contaminate the sample. All individual constraints are easily incorporated as a simple filter in the candidate generation phase.

Table 1 Average solution statistics for instances with 1,000 random sequences of length 10,000, redundancy 1, and number of source sequences varying from 1,000 down to 1.

#Source Seq.	1000	50	10	5	4	3	2	1
#Candidates ($\times 10^3$)	7213.6	1438.6	402.7	225.9	186.9	146.1	102.8	55.7
#Matches ($\times 10^6$)	55.7	35.2	23.2	18.4	16.9	15.0	12.5	8.7
Gen. time	132.3	44.7	35.5	31.4	31.3	30.6	28.1	24.9
Selection time	31.7	10.7	5.3	3.6	3.4	3.1	2.3	1.6
#Distinguishers	14.1	14.1	14.1	14.1	14.0	14.1	14.2	14.5

Redundancy constraints and minimum edit distance constraints. In practice, robust identification requires redundant distinguishability, i.e., more than one distinguisher distinguishing any given pair of genomic sequences. One may also impose a lower bound on the edit distance between any pair of selected distinguishers [15]. Taking into account redundancy requirements is done by maintaining the number of times each pair of genomic sequences has been distinguished. In order to incorporate the minimum edit distance constraint, after selecting a distinguisher we eliminate from consideration all candidates that are within an edit distance smaller than the given threshold.

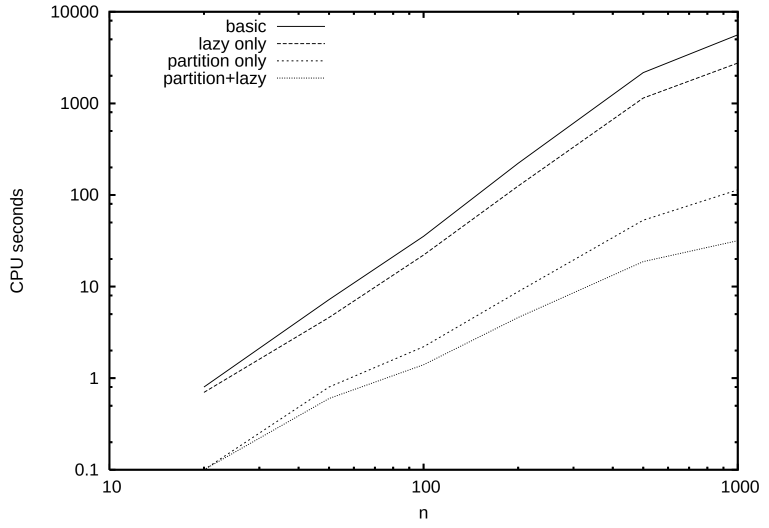
5 Experimental Results

We performed experiments on both randomly generated instances and NCBI databases. Random testcases were generated from the uniform distribution induced by assigning equal probabilities to each of the four nucleotides; these testcases do not contain any nucleotides with degeneracy greater than 1. We also used several testcases consisting of sequences extracted from the NCBI databases [7] as described in Section 5.3. 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 our sequential algorithms.

5.1 Algorithm Scalability

As described in Section 3, there are two main phases in the algorithm: candidate distinguisher generation, and greedy candidate selection. Figure 2 gives 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.

Figure 2 Candidate selection CPU time (in seconds) for n random sequences of length 10,000 and redundancy 1, averaged over 10 instances of each size.



As mentioned in Section 3, a further speed-up technique is to generate candidate distinguishers only from a small number of "source" input sequences. Table 1 gives the average number of candidates, number of matches, runtimes for candidate generation and greedy selection, and number of selected distinguishers for instances with 1,000 random sequences of length 10,000 and redundancy 1, when the number of source sequences is varied from 1,000 down to 1 (the source sequences were chosen at random). Although this speed-up can potentially affect solution quality, we found that on large instances the solution quality loss is minimal even when we generate candidates based on a single input sequence; this case corresponds to pre-assigning a barcode of all 1's to the source 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. As shown in Table 2, this makes the method applicable to hundreds of sequences of bacterial genome size on a well-equipped workstation.

Even when a single input sequence is used to generate candidate distinguishers, this will still result in millions of candidates that must be evaluated by the greedy algorithm for whole-genome barcoding. While our implementation of the setcover greedy algorithm can efficiently handle millions of candidates (Table 2), this may be impractical for other barcoding algorithms. As a more extreme speed-up technique, Cazalis et al. [4] proposed using only a small number (2,000 in [4]) of random candidates in conjunction with various barcoding algorithms including greedy, simulated annealing, and genetic algorithms. However, Cazalis et al. did not provide any data on

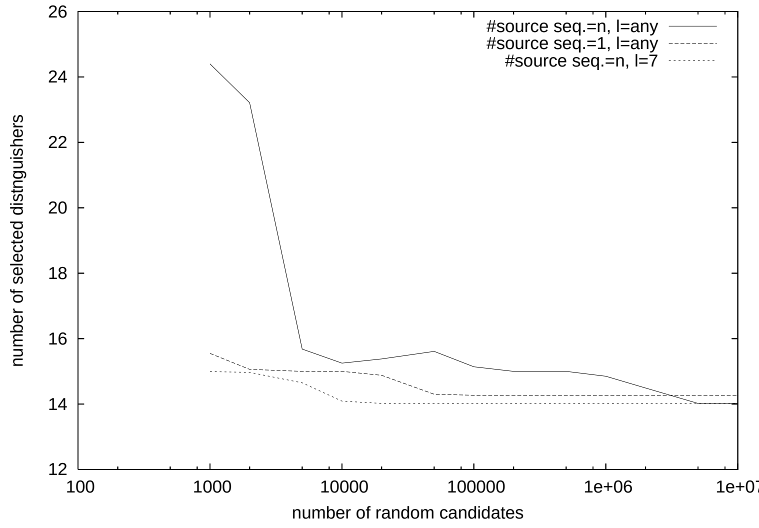
Table 2 Average solution statistics for instances with up to 100 random sequences of length 1,000,000 and redundancy 1 (number of source sequences set to 1).

n	#Candidates	#Matches	Gen. time	Select time	#Dist.
10	2039766.8	8281127.2	45.1	0.6	4.0
20	2607128.9	16730749.0	87.0	1.3	5.0
30	2940246.3	25475766.3	129.1	1.6	5.0
40	3178773.8	34529068.3	172.2	2.6	6.0
50	3363016.8	43802244.9	216.1	3.6	6.6
60	3512271.5	53216933.1	262.7	4.7	7.0
70	3637129.4	62714814.3	303.9	5.1	7.0
80	3744452.1	72256768.1	347.4	6.3	7.4
90	3838282.2	81807129.2	395.5	8.0	8.0
100	3921359.6	91346850.3	444.4	8.5	8.0

the possible solution quality loss from such extreme reductions in the number of candidates, and did not evaluate the relative merits of alternative strategies for sampling these candidates. In Figure 3 we plot the number of distinguishers selected by the greedy setcover algorithm when run on a random subset of all possible candidates, under 3 different candidate sampling strategies: (a) from all source sequences, without length restrictions; (b) from a single random source sequence, without length restrictions; and (c) from all source sequences, with length restricted to 7. Length 7 was chosen here since it leads to the smallest number of selected distinguishers among all fixed distinguisher lengths for instances consisting of 1,000 random sequences of length 10,000 such as those used in this experiment. We note that, although Cazalis et al. [4] suggest using distinguishers of length $\approx \log_2 n$ for a set of n sequences, this rule must be followed with caution. In general, a "most informative" distinguisher is one that appears in exactly half of the sequences, and the typical length of distinguishers with this property depends not only on the number of sequences, but also on their length.

Figure 3 shows that even a few tens of thousands of random candidates sampled using scenarios (b) and (c) above lead to a solution quality very close to that obtained by the setcover greedy algorithm when run on all possible candidates. A much larger number of candidates is required to achieve similar solution quality under scenario (c), i.e., when sampling the random candidates from all sequences and without length constraints. This finding can be explained by the fact that "most informative" candidates represent only a small fraction of the entire set of candidates, while they are more densely represented in the sets of candidates sampled under the first two scenarios.

Figure 3 Number of distinguishers selected by the greedy setcover algorithm from a random subset of all possible candidates. Candidates are randomly chosen (a) from all source sequences, without length restrictions; (b) from a single random source sequence, without length restrictions; and (c) from all source sequences, with length restricted to 7. Each data point represents the average over 100 instances, each consisting of 1,000 random sequences of length 10,000. Redundancy was set to 1 in these experiments.



5.2 *Solution Quality on Random Data*

Table 3 gives the number of distinguishers returned by the setcover greedy algorithm for redundancy varying between 1 and 20 on between 10 and 1,000 random sequences of length 10,000. For comparison we include in the table the results obtained by the information content heuristic results of [1], as well as the information theoretic lower bound of $\lceil \log_2 n \rceil$ for the case

Table 3 Number of distinguishers returned by the setcover greedy algorithm (SGA) for varying redundancy and number of sequences. For each value of n we report the average over 10 testcases, each consisting of n random sequences of length 10,000. For comparison we include information content heuristic results (ICH) and the information theoretic lower bound of $\lceil \log_2 n \rceil$ for redundancy 1 (LB).

Algorithm	r	$n = 10$	$n = 20$	$n = 50$	$n = 100$	$n = 200$	$n = 500$	$n = 1000$
LB	1	4	5	6	7	8	9	10
ICH	1	4.0	5.0	7.0	8.0	10.0	12.2	14.1
SGA	1	4.0	5.0	7.0	8.0	10.0	12.3	14.1
SGA	2	6.7	8.3	10.6	12.5	14.1	16.7	18.9
SGA	3	8.8	11.6	13.6	15.5	17.3	20.1	22.4
SGA	4	10.8	14.0	16.5	18.7	20.7	23.5	26.1
SGA	5	13.6	16.6	19.5	21.5	23.7	26.8	29.5
SGA	10	22.5	26.8	32.0	34.6	37.5	41.7	44.9
SGA	20	43.0	47.6	55.6	59.5	63.4	68.0	72.6

Table 4 Number of distinguishers returned by the setcover greedy algorithm (SGA) and the multi-step rounding algorithm in [2] (RND) for varying redundancy and number of sequences. For each value of n we report the average over 10 testcases, each consisting of n random sequences of length 1,000. Boldface entries correspond to instances for which the multi-step rounding algorithm has better solution quality than setcover greedy.

Algorithm	r	$n = 10$	$n = 20$	$n = 50$	$n = 100$	$n = 200$
SGA	1	4.0	5.0	7.0	9.0	11.0
RND	1	5.0	6.8	10.5	13.0	16.0
SGA	2	6.3	8.2	11.2	12.9	15.0
RND	2	7.3	10.7	14.8	17.0	20.4
SGA	5	13.2	16.1	19.5	22.4	24.6
RND	5	13.2	18.2	23.5	27.3	31.2
SGA	10	22.8	27.0	32.1	36.1	39.4
RND	10	**20.2**	30.9	37.4	41.9	48.3
SGA	20	43.4	48.8	57.0	61.0	65.8
RND	20	**38.9**	50.7	62.6	69.4	76.2
SGA	50	100.9	112.0	125.6	133.8	142.0
RND	50	**92.6**	**107.8**	**125.2**	141.6	159.5
SGA	100	195.0	217.2	239.0	255.5	264.0
RND	100	**184.9**	**205.2**	**236.0**	270.0	289.0
SGA	200	392.00	432.30	471.70	495.40	512.40
RND	200	**372.10**	**412.00**	**455.40**	**485.80**	539.40
SGA	300	594.60	661.30	713.70	744.10	762.00
RND	300	**571.30**	**633.10**	**693.80**	**726.10**	**757.70**

when the redundancy requirement is 1. We note that the number of distinguishers returned by the setcover greedy algorithm is virtually identical to that returned by the information content heuristic, despite the latter one having a better approximation guarantee [1]. Furthermore, the results for redundancy one are 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.

We also compared our setcover greedy algorithm with a recently proposed multi-step rounding algorithm for set multicover [2]. The rounding algorithm has the following steps:

1. Solve the fractional relaxation of the natural integer program formulation of problem [15] (we used the commercial solver CPLEX 9.0 for implementing this step)

2. Scale the fractional solution by an appropriate constant factor (see [2] for details)

3. Deterministically select all distinguishers with a scaled fractional value

exceeding 1

4. Randomly select a subset of the remaining candidates, each candidate being chosen with a probability equal to the scaled fractional value

5. Finally, if the selected set of distinguishers is not yet feasible, add further distinguishers using the setcover greedy algorithm

The approximation guarantee established in [2] for the general set multicover problem translates into an approximation factor of $2\ln n - \ln r$ for robust string barcoding with redundancy r, which suggests that the multi-step rounding algorithm is likely to improve upon the setcover greedy for high redundancy constraints. Table 4 gives the results of experiments comparing the setcover greedy and multi-step rounding algorithms on testcases consisting of up to 200 random sequences, each of length 1,000 for redundancy requirement ranging from 1 to 300. The results confirm that the multi-step rounding algorithm has better solution quality than setcover greedy when redundancy requirement is large relative to the number of sequences (entries typeset in boldface), yet the setcover greedy has best performance for most practical redundancy requirements.

5.3 Experiments on Genomic Data

In a first set of experiments we used 10 groups of testcases obtained from the authors of [15], each consisting of random sets of viruses, respectively HIV strains, extracted from GenBank. Most of these testcases contain a small number of degenerate bases; detailed testcase parameters are given in Table 5. Hence, we cannot use the partition method for computing the number of sequence pairs distinguished by a candidate in the greedy selection phase, and we have to use the slower matrix datastructure. Table 6 gives the average runtime and the number of distinguishers selected by the setcover greedy algorithm on these testcases when using all available candidates. For comparison, we also include the average number of distinguishers obtained in [15] by solving an integer program formulation of the problem using the CPLEX commercial optimization package. However, the results in [15] may be overly optimistic since the underlying integer program treats degenerate bases as distinct nucleotides. (We do not know if degenerate bases were actually used in distinguishers selected by CPLEX since we do not have access to the solutions in [15].) With few exceptions, the greedy algorithm comes very close to the solution computed by the integer program.

In a second set of experiments we ran our algorithm on a set of 29 complete microbial genomic sequences extracted from NCBI databases [7]. 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). Unlike random

Table 5 Size and algorithm parameters for genomic instances from [15].

Test Group	#Test cases	Avg. n	Avg. str. len.	Avg. #degen.	l_{min}	l_{max}	Min edit	r
hiv0	27	91.44	967.50	59.81	15	40	4	5
hiv1	26	89.28	684.91	53.19	15	40	4	2
hiv4	26	90.80	723.47	41.27	15	40	2	2
hiv5	26	90.40	1085.01	35.50	15	40	2	5
hiv6	26	90.92	849.47	45.77	15	40	4	5
len0	26	105.40	1086.28	36.27	17	21	4	5
s0	26	51.12	1123.17	54.27	15	40	4	5
s1	26	70.64	942.19	18.69	15	40	4	5
s2	26	105.96	897.63	29.96	15	40	4	5
s3	26	129.92	948.56	32.87	15	40	4	5

Table 6 Average solution statistics for genomic instances from [15].

Test Group	#Candidates	#Matches	Gen. time	Select time	SGA #Dist.	ILP #Dist.
hiv0	175707.8	440615.7	7.3	100.0	137.8	89.44
hiv1	158530.0	396909.4	4.4	43.0	70.8	45.12
hiv4	125694.3	333881.9	4.8	35.4	71.8	43.88
hiv5	146462.4	377735.3	8.3	104.8	177.0	132.76
hiv6	147135.1	388387.0	5.6	98.4	167.8	126.61
len0	42091.0	175841.9	3.4	26.7	180.6	160.29
s0	282467.7	726758.6	5.3	167.6	108.0	99.92
s1	123694.9	452387.2	5.5	65.1	126.8	117.20
s2	194253.1	755897.3	7.3	161.1	178.1	115.70
s3	278795.2	1075451.6	10.7	308.7	216.6	200.91

Table 7 Results on a set of 29 NCBI complete microbial genomes. Candidate generation time is approximately 335 seconds for all combinations of parameters.

Redundancy	l_{min}	l_{max}	MinEdit	Select time	#Distinguishers
1	0	∞	0	14.2	6.0
1	15	40	6	2.6	8.0
5	0	∞	0	20.3	21.0
5	15	40	6	8.7	31.0
10	0	∞	0	22.9	41.0
10	15	40	6	16.4	60.0
20	0	∞	0	26.8	76.0
20	15	40	6	33.4	123.0

testcases, the sequences in the NCBI data set contain a small number of degenerate bases, 861 bases in total. Therefore, we cannot use the partition method for computing the number of sequence pairs distinguished by a candidate in the greedy selection phase, and we have to use the slower matrix datastructure. 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 we 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 [15]. In all experiments, we generated candidates based only on the shortest sequence of 490 Kbases.

The results on this NCBI dataset are given in Table 7. 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).

6 Conclusions

In this paper we have given highly scalable algorithms for the robust string barcoding problem, and have shown that distinguisher selection based whole genomic sequences results in a number of distinguishers nearly matching the information theoretic lower bounds for the problem.

In ongoing work we are exploring heuristics and approximation algorithms for several extensions of the string barcoding problem. First, we are considering the use of probe mixtures as distinguishers. With most microarray technologies it is feasible to spot/synthesize a mixture of oligonucleotides at any given microarray location. The DNA of a pathogen will hybridize to such a location if it contains at least one substring which is the Watson-Crick complement of one of the oligonucleotides in the mixture. Using oligonucleotide mixtures as distinguishers can reduce the number of spots on the array – and therefore barcode length – closer to the information theoretical lower-bound of $\log_2 n$. The reduction promises to be particularly significant when reliable hybridization requires relatively long distinguishers; in these cases even the optimum barcoding length is far from $\log_2 n$ [15]. A special case of this approach is the use of *degenerate* distinguishers similar to the degenerate primers that have been recently employed in multiplex PCR amplification [12, 16]. Degenerate distinguishers are particularly attractive for string barcoding since their synthesis cost is nearly identical to the synthesis cost of a single non-degenerate distinguisher (synthesis requires the same number of steps, the only difference is that multiple nucleotides must be added in some of the synthesis steps).

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, see, e.g., [9] for studies in detecting more than one HPV (human papiloma virus) genotype with varying rate of multiple HPV infections carried by the same HPV carrier. In future work we plan to develop extensions of the barcoding technique that can reliably detect multiple pathogens for a given bound on the number of pathogens present.

Acknowledgments

The authors would like to thank Claudia Prajescu for her help with the implementation of the multi-step rounding algorithm in [2].

References and Notes

1 Berman, B., DasGupta, B. and Kao, M.-Y. (2004) 'Tight approximability results for test set problems in bioinformatics', *Proc. 9th Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 3111 of *Lecture Notes in Computer Science*, pages 39–50, Springer-Verlag, Berlin.

2 Berman, P., DasGupta, B. and Sontag, E. (2004) 'Randomized approximation algorithms for set multicover problems with applications to reverse engineering of protein and gene networks', *Proc. 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 2748 of *Lecture Notes in Computer Science*, pages 39–50, Springer-Verlag, Berlin. (Also to appear in a special issue on computational biology of *Discrete Applied Mathematics.*).

3 Borneman, J., Chrobak, M., Vedova, G.D., Figueora, A. and Jiang, T. (2001) 'Probe selection algorithms with applications in the analysis of microbial communities', *Bioinformatics*, Vol. 1, pp. 1–9.

4 Cazalis, D., Milledge, T. and Narasimhan, G. (2004) 'Probe selection problem: Structure and algorithms', *Proc. 8th Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2004)*, pp. 124–129.

5 Cheung, V.G. and Nelson, S.F. (1996) 'Whole genome amplification using a degenerate oligonucleotide primer allows hundreds of genotypes to be performed on less than one nanogram of genomic DNA', *Proc. Natl. Acad. Sci. USA.*, Vol. 93, pp. 14676–14679.

6 Chvátal, V. (1979) 'A greedy heuristic for the set covering problem', *Math. of Op. Res.*, Vol. 4, pp. 233–235.

7 NCBI (2004) *Completed Microbial Genomes*, http://www.ncbi.nlm.nih.gov/genomes/microbes/complete.html.

8 Dean, F.B. et al. (2002) 'Comprehensive human genome amplification using multiple displacement amplification', *Proc. Natl. Acad. Sci. USA.*, Vol. 99, pp. 5261–5266.

9 Gharizadeh, B. et al. (2003) 'Viral and microbial genotyping by a combination of multiplex competitive hybridization and specific extension followed by hybridization to generic tag arrays', *Nucleic Acids Research*, Vol. 31, No. 22, pp. e146.

10 Guibas, L.J. and Odlyzko, A.M. (1981) 'String overlaps, pattern matching, and nontransitive games', *Journal of Combinatorial Theory Series A*, Vol. 30, pp. 183–208.

11 Johnson, D.S. (1974) 'Approximation algorithms for combinatorial problems', *J. Comput. Sys. Sci.*, Vol. 9, pp. 256–278.

12 Linhart, C. and Shamir, R. (2002) 'The degenerate primer design problem', *Bioinformatics*, Vol. 18, pp. S172–S181.

13 Lovász, L. (1975) 'On the ratio of optimal integral and fractional covers', *Discrete Mathematics*, Vol. 13, pp. 383–390.

14 Odlyzko, A.M. (1995) 'Asymptotic enumeration methods', R.L. Graham, M. Grötschel, and L. Lovász (eds.), *Handbook of Combinatorics, Vol. II*, pp. 1063–1230, MIT Press, Cambridge, MA.

15 Rash, S. and Gusfield, D. (2002) 'String barcoding: Uncovering optimal virus signatures', *Proc. 6th Annual International Conference on Computational Biology*, pp. 254–261.

16 Souvenir, R., Buhler, J., Stormo, G. and Zhang, Z. (2003) 'Selecting degenerate multiplex PCR primers', *Proc. 3rd Intl. Workshop on Algorithms in Bioinformatics (WABI)*, pp. 512–526.

17 Vazirani, V.V. (2001) *Approximation Algorithms*, Springer-Verlag, Berlin.