

# Genome Identification and Classification by Short Oligo Arrays

Stanislav Angelov<sup>1\*</sup>, Boulos Harb<sup>1\*\*</sup>, Sampath Kannan<sup>1\*\*\*</sup>, Sanjeev Khanna<sup>1†</sup>, Junhyong Kim<sup>2‡</sup>, and Li-San Wang<sup>2§</sup>

<sup>1</sup> Department of Computer and Information Science,  
School of Engineering and Applied Sciences,  
University of Pennsylvania, Philadelphia, PA 19104, USA  
{angelov, boulos, kannan, sanjeev}@cis.upenn.edu  
<sup>2</sup> Department of Biology, School of Arts and Sciences,  
University of Pennsylvania, Philadelphia, PA 19104, USA  
junhyong@sas.upenn.edu, lswang@mail.med.upenn.edu

**Abstract.** We explore the problem of designing oligonucleotides that help locate organisms along a known phylogenetic tree. We develop a suffix-tree based algorithm to find such short sequences efficiently. Our algorithm requires  $O(Nm)$  time and  $O(N)$  space in the worst case where  $m$  is the number of the genomes classified by the phylogeny and  $N$  is their total length. We implemented our algorithm and used it to find these discriminating sequences in both small and large phylogenies. We believe our algorithm will have wide applications including: high-throughput classification and identification, oligo array design optimally differentiating genes in gene families, and markers for closely related strains and populations. It will also have scientific significance as a new way to assess the confidence in a given classification.

## 1 Introduction

Short sequence tags, typically oligonucleotides of 20–100 base pairs (bp), have been widely used in genomic technologies, e.g. the Affymetrix GeneChip; SAGE [1], EST [2], and STS [3]. The idea is that a sequence tag or a set of tags will be a unique subsequence of a longer target sequence and, therefore, the tag(s) will serve to identify the target sequence. In the limit, the tags may represent the whole genome or a set of genomes. More recently, short sequence tags have been

---

\* Supported in part by NIGMS 1-P20-GM-6912-1 and NSF ITR 0205456

\*\* Supported in part by NIH Training Grant T32HG0046

\*\*\* Supported in part by ARO grant DAAD 19-01-1-0473 and NSF grant CCR-0105337

† Supported in part by Alfred P. Sloan Research Fellowship, NSF Career Award CCR-0093117, and NSF ITR 0205456

‡ Supported in part by NIGMS 1-P20-GM-6912-1

§ Supported in part by NIH Training Grant in Cancer and Immunopathobiology (1 T32 CA101968)

proposed as IDs for whole organisms [4], with the related idea that some reasonably long stretch of homologous sequence can be used as sufficient information for taxonomy. In this application, the tags are assayed not only for presence and absence but information about their string difference is also utilized in the taxonomic procedures. With shorter oligo tags, the string differences between different tags carry little information. In addition, string difference cannot be easily assayed by high-throughput hybridization based technologies. Therefore, with short oligo array technique the information is contained in the configuration of presence and absence of the suite of oligonucleotides assayed (called *probes*) by the hybridization array. Hypothetically, if the set of probes is very large and exhaustively assays a sufficiently large address space, say, the set of all 20 base pair strings, we can use the presence/absence information to construct or recover detailed taxonomic or phylogenetic information. However, practical devices are limited to assaying on the order of  $10^4$  to  $10^5$  different probes. Fortunately, genomes are not random collection of strings but they are related to each other by phylogenetic relationships, thus an efficient set of probes can be designed to recover genomic information from a limited but a judicious selection of the address space.

Databases have been set up for organism specific oligonucleotides and primer design (see for example [5], and [6]). Heuristic construction of short sequence tags has been used for identification and classification (reviewed in [7]), but no algorithm has been presented for tag design. Previous computational oligo-sequence design work has mostly concentrated on optimizing hybridization kinetics or assembly into longer fragments (for example, [8], [9], [10], and [11]) but has not considered the problem of optimal design for classifications.

We explore the problem of designing short sequence tags (simply called tags from here on) that help locate organisms along a known phylogenetic or a classification tree with binary splits. (For expedience, we will only refer to phylogenetic trees from here on, however many biologists make distinctions between phylogenetic trees and other classification trees.) Specifically, given complete genomes for a set of organisms and the binary phylogenetic tree that describes their evolution, we develop an efficient algorithm for detecting all *discriminating* tags. We will say that a tag  $t$  is discriminating at some node  $u$  of the phylogeny if all genomes under one branch of  $u$  contain  $t$  while none of the genomes under any other branch contain  $t$ . Thus, a set of discriminating tags for all the nodes of a phylogeny allows us to place a genome in the phylogeny by a series of binary decisions starting from the root. This procedure can be implemented experimentally as a microarray hybridization assay, enabling a rapid determination of an unidentified organism in a predetermined phylogeny or classification.

Using suffix trees [12], we implement an  $O(Nm)$  time algorithm that computes all discriminating tags at all nodes of the phylogeny, where  $N$  is the total length of the genomic sequences under consideration and  $m$  is their number. The suffix tree (as well as its generalization for multiple sequences) is a compact tree representation of all suffixes of a given input text. The algorithm enumerates all substrings of the input via a bottom-up walk of the computed generalized

suffix tree, maintaining the set of input sequences that contain each substring. This information is sufficient to determine for which nodes of the phylogeny the current substring is discriminating. We also analyze special cases of the problem that can be solved in  $O(N \log m)$  time and  $O(N)$  space. Finally, we demonstrate the existence of discriminating tags by running our algorithm on biological data sets whose estimated phylogenies we obtained from [13] and [14].

## 2 Finding Discriminating Tags

Recall that a discriminating tag for a node  $u$  in a given phylogeny is a substring that is present in every genome under one branch of  $u$  and is not present in any genome under all other branches of  $u$ . In this section, we outline our basic algorithm for finding discriminating tags, and we present an enhancement to reduce the algorithm’s space requirement. Both the algorithm, and its enhancement build on the solution to the multiple common substring problem described in [12].

### 2.1 Preliminaries

We state the problem more precisely as follows. Let  $S = \{s_1, \dots, s_m\}$  be a set of  $m$  strings, also called *species*, and let  $P$  be a phylogeny represented by a rooted binary tree with  $m$  leaves labeled by  $S$ . For each internal node  $u$  of  $P$ , we wish to find the set of discriminating tags for  $u$ . Without loss of generality, we will focus on finding discriminating tags present in the left subtree of every node. We assume that the strings are drawn from a bounded-size alphabet  $\Sigma$ , and that  $\sum_{i=1}^m |s_i| = N$  where  $|s_i|$  denotes the number of characters in the string  $s_i$ .

**Suffix Trees.** Suffix trees, first introduced by [15], play a central role in our algorithms. A suffix tree is a rooted directed tree that compactly represents all suffixes of a string. Each edge in a suffix tree for a string  $s$  is labeled with a non-empty substring of  $s$ . We adopt the following definition from [12]. The *path-label* of a node  $v$  in a suffix tree  $T$  for  $s$  is the string formed by following the path from the root of  $T$  to  $v$ . We will denote it by  $path(v)$ . The path-labels of the  $|s|$  leaves of  $T$  spell out the suffixes of  $s$ , and the path-labels of internal nodes spell out substrings of  $s$ . Furthermore, the suffix tree ensures that there is a unique path from the root, not necessarily ending at an internal node, that represents each substring of  $s$  (see Fig. 1(a)). The time and space requirements of building a suffix tree are linear in the size of the string it represents [15–17].

Our algorithms, more specifically, are based on *generalized suffix trees*. A generalized suffix tree extends the idea of a suffix tree for a single string to a suffix tree for a set of strings. As described in [12] such a tree can conceptually be constructed by appending a unique terminating marker not in  $\Sigma$  to each string in the given set, concatenating the strings, and building a suffix tree for the resulting string. The tree is post-processed so that each path-label of a leaf in the tree spells a suffix of one of the strings in the set and, hence, is terminated



Each  $B(v)$  is represented by a  $m$ -bit vector  $b_v$  where  $b_v[i] = 1$  iff  $s_i \in S$ . We compute all the bit vectors by performing a post-order walk on  $T$  in  $O(N)$  time. Since each bit-vector participates in one union operation, the running time of this phase is  $O(Nm)$ . Clearly,  $O(Nm)$  space is required to store all the bit vectors.

2. Compute the discriminating tags for each  $u$  of  $P$ .

Observe that the set  $B(v)$  for  $v \in T$  contains sufficient information to determine if  $path(v)$  is a discriminating tag for any  $u \in P$ . We determine all the nodes of  $P$  for which  $path(v)$  is a discriminating tag efficiently as follows:

Assume *w.l.o.g.* that the leaves of  $P$  are labeled with  $s_1, s_2, \dots, s_m$  from left to right. We first label each internal node of  $P$  with the index  $i$  of the rightmost string  $s_i$  in its left subtree. This labeling is unique since we can assume that each internal node has exactly 2 children. We then compute the number of leaves in both the left and right subtrees of each node. Denote these two quantities for  $i \in P$  by  $left-size(i)$  and  $right-size(i)$  respectively. Now,  $path(v)$  is a discriminating tag for  $i$  if the last run of 1's in  $b_v$  up to and including  $i$  is at least  $left-size(i)$  and  $b_v[i]$  is followed by at least  $right-size(i)$  0's. Note that if  $path(v)$  is discriminating for  $i \in P$ , then the prefixes of  $path(v)$  that are not prefixes of  $v$ 's parent are also discriminating for  $i$ . Call these tags the discriminating tags *induced* by  $v$ , and note that they are determined by the label of the edge  $(parent(v), v)$  in  $T$ .

We can compute the left-to-right running count of 1's, and the right-to-left running count of 0's for each  $b_v$  in  $O(m)$  time. Since there are  $N$  bit vectors, the process requires  $O(Nm)$  time.

**Remark.** If all input strings have length  $O(n)$  such that  $N = O(nm)$ , then we need only store  $O(n)$  bit vectors at a time, reducing the space requirement to  $O(N)$ .

### 2.3 Improving the Time and Space Requirement

We can improve the running time and space requirement of the algorithm above by slightly modifying the linear time algorithm used to solve the multiple common substring problem introduced in [18]. The algorithm computes  $C_S(v) = |B(v)|$ , the number of species in  $S$  containing  $path(v)$ , for all  $v$  of  $T$  in  $O(N)$  time and space. It achieves this running time by utilizing suffix trees and constant-time *lowest common ancestor (lca)* queries [19,20]. The *lca* of two nodes  $v$  and  $v'$  in a rooted tree is defined to be the deepest node in the tree that is an ancestor of both  $v$  and  $v'$ . Observe that  $C_S(v)$  is equal to the number of *distinct* string markers terminating the path-labels of the leaves under  $v$ . The total number of string markers in  $v$ 's subtree can be computed via a post-order walk of  $T$ . However, this number will be an over-count of  $C_S(v)$  if there are duplicate markers under  $v$ . Suppose  $l$  and  $l'$  are two leaves such that  $l'$  is the first leaf

with the same marker as  $l$  that the walk encounters after  $l$ . We then know that if  $v$  is an ancestor of  $lca(l, l')$ , the walk will over-count  $C_S(v)$  by one due to  $l$  and  $l'$ . Now, for each such pair  $l$  and  $l'$ , the algorithm first increments a counter maintained at  $lca(l, l')$ . Then, using the counters, it determines the number of duplicate markers under each node in a bottom-up fashion.

Our modification is straightforward, and we demonstrate it for the root of our phylogeny  $P$ . Suppose that the root is labeled  $r$ . Then, a tag is discriminating for  $r$  if it is a substring of all the strings of  $L = \{s_1, \dots, s_r\}$  and none of the strings of  $R = \{s_{r+1}, \dots, s_m\}$ . Instead of just maintaining one counter  $C_S(v)$  for each  $v \in T$ , we maintain  $C_L(v)$  and  $C_R(v)$ , the number of strings of  $L$  and  $R$  containing  $path(v)$  respectively. Now,  $path(v)$  is discriminating for  $r$  if  $C_L(v) = |L|$  and  $C_R(v) = 0$ .

The running time of this procedure is optimal for computing the discriminating tags for one node of  $P$ : It is linear in the total length of the strings in the subtree rooted at the node. Now, for  $u \in P$ , let  $S_u \subseteq S$  denote those strings in  $u$ 's subtree. The running time for computing the discriminating tags for all the nodes of  $P$  is then:

$$\sum_{u \in P} \sum_{s \in S_u} |s| = \sum_{s \in S} \sum_{u \in P: s \in S_u} |s| = O(N \cdot depth(P)) ,$$

which is  $O(N \log m)$  for a balanced phylogeny and  $O(Nm)$  in the worst case. However, the space requirement is  $O(N)$ .

## 2.4 Minimal and Maximal Tags

We now consider the problem of finding *minimal* and *maximal* discriminating tags. A discriminating tag is said to be minimal if no substring of it is also discriminating. Correspondingly, a discriminating tag is said to be maximal if it is not a substring of another discriminating tag. We are interested in finding minimal and maximal tags for mainly two reasons. First, as implied by the claim below, the set of discriminating tags can potentially be enormous; hence, finding the minimal and maximal tags can be viewed as a data reduction step. More importantly, however, are the limitations on the lengths of tags that can be used in hybridization arrays. For example, tags used in the Affymetrix oligonucleotide microarrays are 25 bp long [21, pp. 3-4], whereas tags used in spotting arrays are between 100 and 5000 bp long [22, p. 11]. Consequently, the length distribution of minimal and maximal tags indicates the feasibility of executing our discrimination scheme in actual hybridization assays.

We start with a claim that will help us efficiently identify the subsets of minimal and maximal tags for a node in the phylogeny after computing the discriminating tags for the node.

*Claim.* Let  $t$  and  $t'$  be discriminating tags for some node  $u \in P$ , where  $t'$  is a substring of  $t$ . Then, any string  $t''$  such that  $t'$  is a substring of  $t''$  and  $t''$  is a substring of  $t$  is also a discriminating tag for  $u$ .

*Proof.* Since  $t'$  is a substring of  $t$ , it is present in all the sequences in the left subtree of  $u$ . Further, since  $t'$  is a substring of  $t''$  and it is not present in any sequence in the right subtree of  $u$ ,  $t''$  will not be present in any of these sequences.

**Corollary 1.** *A discriminating tag is minimal if and only if its longest proper prefix and suffix are not discriminating.*

**Corollary 2.** *A discriminating tag is not maximal if and only if it is the longest proper prefix or suffix of a discriminating tag.*

Recall that in a suffix tree, the path-label of a node  $v$ ,  $path(v)$  is a prefix of  $path(u) : u \in child(v)$  and that  $path(parent(v))$  is a prefix of  $path(v)$ . We can also quickly determine the corresponding suffix relationships for  $path(v)$  via the *suffix links* in the tree. The suffix link of an internal node  $v$  in the suffix tree is a directed non-tree edge from  $v$  to the longest proper suffix of  $path(v)$ . For example, if  $path(v) = a\beta$  where  $a \in \Sigma$  and  $\beta \in \Sigma^*$ , then  $(v, v') : path(v') = \beta$  will be the suffix link of  $v$ . By construction, every internal node of the suffix tree has a suffix link [17]. As an example, the non-tree edge  $(v, v')$  in Fig. 1(a) is a suffix link.

Our task is then simple. Suppose  $t$  is a discriminating tag induced by  $v$ . In order for  $t$  to be minimal, it should be the shortest discriminating tag induced by  $v$ ;  $path(parent(v))$  should not be discriminating since it is the longest prefix of  $t$ ; and,  $path(u) : (v, u)$  is a suffix link should not be discriminating since otherwise the longest proper suffix of  $t$  would be discriminating. In order for  $t$  to be maximal, on the other hand,  $t$  must equal  $path(v)$ , the path labels of the children of  $v$  should not be discriminating; and,  $path(u)$  should not be discriminating if  $(u, v)$  is a suffix link.

Conceptually, we can ensure that all tags are induced by internal nodes for which suffix links are available by duplicating each input string and assigning each copy a distinct terminating character. Finally, the outlined computation can be readily incorporated in the algorithms above with no effect on the asymptotic running time and space requirements.

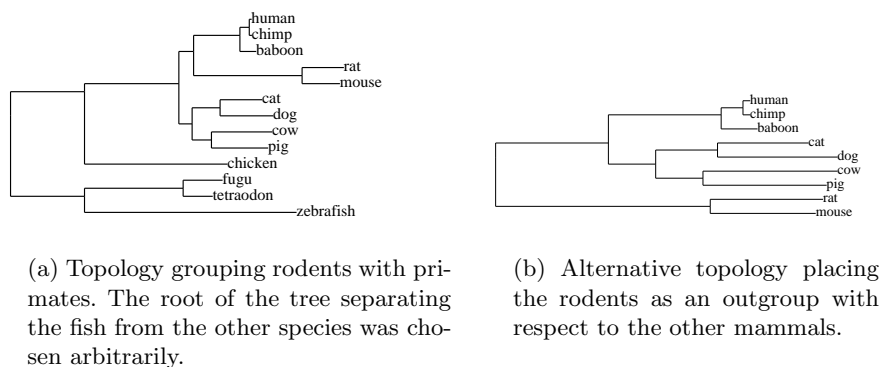
### 3 Experimental Results

As proof of concept, we first ran our algorithm on the estimated phylogeny for the 13 species: human, baboon, chimpanzee, rat, mouse, cat, dog, cow, pig, chicken, fugu, tetraodon, and zebrafish [13]. The phylogeny (Fig. 2(a)) is deduced from the genomic region orthologous to a segment of about 1.8 megabases (Mb) on human chromosome 7 containing the Cystic Fibrosis Transmembrane Conductance Regulator (CFTR) gene and 9 other genes. We will refer to this as the *CFTR data set*.

We also produced discriminating tags for some of the phylogenetic trees and the corresponding small subunit (SSU) prokaryotic ribosomal RNA sequences obtained from the Ribosomal Database Project (RDP) [14].

### 3.1 The CFTR Data Set

The CFTR data set we used to perform our first set of experiments consists of 13 sequences<sup>3</sup> of total length greater than 14Mb. The estimated phylogeny, as well as an alternative grouping of the mammals, are shown in Fig. 2 [13].



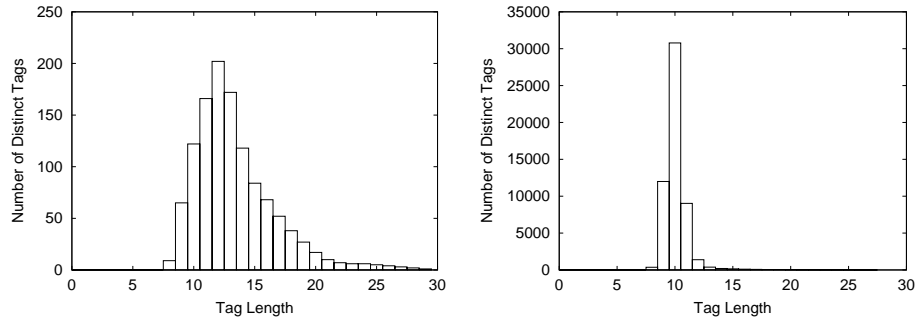
**Fig. 2.** Maximum-likelihood phylogenies for the CFTR data set.

Our algorithm produced discriminating tags for each node in the phylogeny of the CFTR data set. Figure 3 plots the distribution of discriminating tag lengths for two nodes. We also produced the discriminating tags for the phylogeny shown in Fig. 2(b). The results for the root are shown in Fig. 4. Note that figures 3(b) and 4 show discriminating tags for the two alternative high-level partitions of the mammals.

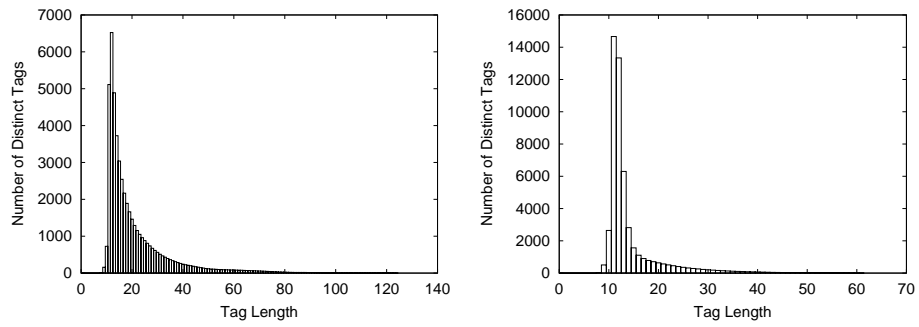
**Random Leaf Permutation.** As a demonstration that the discriminating tags found are in fact dependent on the classification, we compared the distribution of the tags induced by the estimated phylogeny with the distribution of the tags induced by a random permutation of the sequences at the leaf nodes. The latter essentially represents a null hypothesis. Our results show that the number of discriminating tags generated in the latter case significantly decreases. For example, at the root node of the phylogeny for our 13 species, we only found two discriminating tags for the right branch as opposed to nearly 55,000 tags we discovered when using the original phylogeny.

**Random Tags from Conserved Regions.** We examine the selectivity of our procedure by testing whether random substrings drawn from sequences' exon

<sup>3</sup> <http://www.nisc.nih.gov/data>

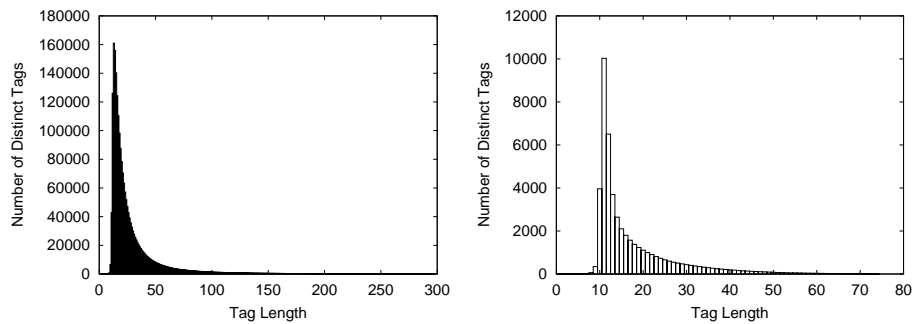


(a) Discriminating tags for the root separating the fish from the other 10 species.



(b) Discriminating tags for the node separating the artiodactyls and carnivores from the rodents and primates.

**Fig. 3.** Length distribution of discriminating tags for two nodes of the CFTR phylogeny in Fig. 2(a). The left (right) panel displays discriminating tags present in the left (right) subtree of the corresponding node.



**Fig. 4.** Length distribution of discriminating tags for the root of the alternative grouping of the mammals shown in Fig. 2(b).

regions form discriminating tags or not. We obtained the annotations for the CFTR data set from the NISC Comparative Sequence Program website<sup>3</sup>, and performed this experiment for the root of the phylogeny in Fig. 2(a). We observed that random substrings of such highly conserved regions of length more than 15, drawn from random sequences under one branch of the root, are rarely contained in any of the sequences under the other branch; however, each species from the same branch contained only a small fraction of the substrings, implying that those random substrings are not discriminating.

### 3.2 Large Phylogenies

In order to test the existence of tags in large phylogenies, we ran our algorithm on some of the phylogenies in the RDP-II database [14]. As an example, Fig. 5 shows the phylogeny for the Crenarch tree. The tree classifies 85 organisms with average sequence length equal to 1448 bp. We found discriminating tags for all nodes except the root of the tree. In general, for all the trees we considered (87–218 organisms), we found discriminating tags for most nodes except those that are close to the root of each tree, which is not surprising given the short sequence lengths of the SSU rRNA.

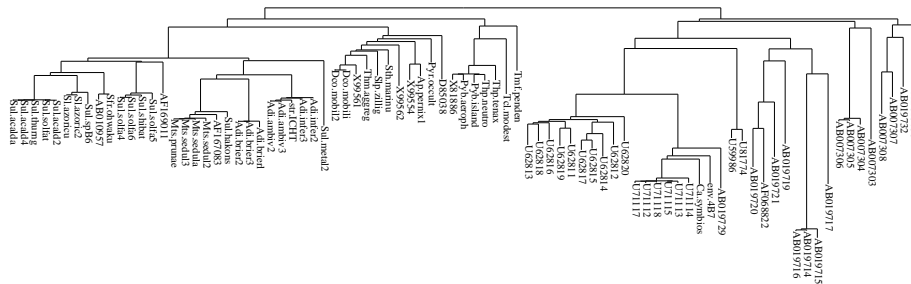


Fig. 5. The Crenarch 85-organism tree

## 4 Discussion and Conclusion

In this paper, we proposed an efficient algorithm to generate short oligonucleotide tags for the purpose of identifying genomes within a preexisting phylogenetic (or classification) tree. We demonstrated tag feasibility using actual biological datasets and showed that typically a large number of useful tags can be obtained for each vertex in the phylogenetic tree. We also showed that the frequency and the lengths of tags depend on the tree structure – i.e., the tree structure that closely reflects the generative process of the targeted sequences yielded the largest collection of tags. There are other aspects of the tags such as the

length of the minimal and maximal tags that depend on the stochastic process of evolution and the phylogenetic tree. Future work will be required to understand the statistical properties of discriminating tags vis-à-vis the phylogenies.

There are now multiple efforts to use modern molecular techniques for understanding the diversity of life (<http://www.nsf.gov/bio/progdes/bioatol.htm>) and cataloging organisms [4]. It is clear that these efforts will not succeed without a significant solution to scaling up both the data collection and the computational analysis. The current algorithm can be directly applied to the problem of obtaining efficient molecular markers (so-called bar-coding) to identify field samples against existing phylogenies or classification trees. However, it differs from some of the bar-coding efforts in that individual nucleotide identity is not required; assaying sequence identity is currently less amenable to high-throughput assays. More generally, we would like to obtain a high-throughput data collection method that will place novel organisms within an existing tree or *de novo* build phylogenetic trees for a large collection of organisms where prior information is available for only a small subset of taxa.

The current algorithm represents the first step in approaching the above goal. There are multiple remaining problems to convert this into a practical solution. First, given a set of discriminating tags, we need additional procedures to select optimal tags in terms of their biochemical characteristics including hybridization kinetics, potential for mismatch vis-à-vis other probes, optimal length, and so on. Second, the total genomic sequence may not be available for many organisms and if we generate tags from partial sequence information (such as the CFTR region above) we would like the tags to be valid for the whole genome. Third, our current procedure concentrates on placing known organisms on known trees, but ideally we would like a procedure that will place novel organisms in their proper locations by phylogenetic or classification criteria. That is, we would like the tags to generalize to novel information both in the tree and in the genome. These problems lead to the fourth problem of simultaneously optimizing the tags in terms of their potential instrument error, their generalization probability, and discriminating sensitivity. Finally, existing phylogenetic methods or classification methods are not based on models of presence/absence of genome-scale oligo tags; additional studies will need to be carried out for the tree reconstruction or recovery methods.

Rapid detection of unknown biological agents or large-scale construction of genomic phylogenies or classifications all demand the development of high-throughput analysis procedures. Many of the existing biochemical techniques are adequate to meet this challenge. The remaining problems are computational, both for efficient instrument design and for subsequent analysis. We believe the class of algorithms studied here and solutions to the problems proposed here will be critical toward designing high-throughput data collection strategies, maximizing efficiency by leveraging prior phylogenetic or taxonomic information.

## References

1. Velculescu, V., Zhang, L., Vogelstein, B., Kinzler, K.: Serial analysis of gene expression. *Science* **270** (1995) 484–487
2. Adams, M., Kelley, J., Gocayne, J., Dubnick, M., Polymeropoulos, M., Xiao, H., C.R. Merrill, e.a.: Complementary DNA sequencing: expressed sequence tags and human genome project. *Science* **252** (1991) 1651–1656
3. Olson, M., Hood, L., Cantor, C., Botstein, D.: A common language for physical mapping of the human genome. *Science* **245** (1989) 1434–1435
4. Hebert, P., Cywinska, A., Ball, S., deWaard, J.: Biological identifications through DNA barcodes. *Proc. of the Royal Society of London* **270** (2003) 313–321
5. Onodera, K., Melcher, U.: Virologo: a database of virus-specific oligonucleotides. *Nucl. Acids. Res.* **30** (2002) 203–204
6. Ashelford, K.E., Weightman, A.J., Fry, J.C.: Primrose: a computer program for generating and estimating the phylogenetic range of 16S rRNA oligonucleotide probes and primers in conjunction with the rdp-ii database. *Nucl. Acids. Res.* **30** (2002) 3481–3489
7. Amann, R., Ludwig, W.: Ribosomal rna-targeted nucleic acid probes for studies in microbial ecology. *FEMS Microbiology Reviews* **24** (2000) 555–565
8. Matveeva, O.V., Shabalina, S.A., Nemtsov, V.A., Tsodikov, A.D., Gesteland, R.F., Atkins, J.F.: Thermodynamic calculations and statistical correlations for oligo-probes design. *Nucl. Acids. Res.* **31** (2003) 4211–4217
9. Kaderali, L., Schliep, A.: Selecting signature oligonucleotides to identify organisms using DNA arrays. *Bioinformatics* **18** (2002) 1340–1349
10. Frieze, A.M., Halldorsson, B.V.: Optimal sequencing by hybridization in rounds. *Journal of Computational Biology* **9** (2002) 355–369
11. Mitsuhashi, M., Cooper, A., Ogura, M., Shinagawa, T., Yano, K., Hosokawa, T.: Oligonucleotide probe design - a new approach. *Nature* **367** (1994) 759–761
12. Gusfield, D.: *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, New York (1997)
13. Thomas, J., et al.: Comparative analyses of multi-species sequences from targeted genomic regions. *Nature* **424** (2003) 788–793
14. Maidak, B.L., Cole, J.R., Lilburn, T.G., Parker, Charles T., J., Saxman, P.R., Farris, R.J., Garrity, G.M., Olsen, G.J., Schmidt, T.M., Tiedje, J.M.: The rdp-ii (ribosomal database project). *Nucl. Acids. Res.* **29** (2001) 173–174
15. Weiner, P.: Linear pattern matching algorithms. In: *Proc. of the 14th IEEE Symposium on Switching and Automata Theory*. (1973) 1–11
16. McCreight, E.M.: A space-economical suffix tree construction algorithm. *Journal of the ACM (JACM)* **23** (1976) 262–272
17. Ukkonen, E.: On-line construction of suffix-trees. *Algorithmica* **14** (1995) 249–260
18. Hui, L.: Color set size problem with applications to string matching. In: *3rd Symposium on Combinatorial Pattern Matching*. Volume 644 of *Lecture Notes in Computer Science*, Springer (1992) 227–240
19. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. *SIAM Journal of Computing* **13** (1984) 338–355
20. Schieber, B., Vishkin, U.: On finding lowest common ancestors: Simplifications and parallelization. *SIAM Journal of Computing* **17** (1988) 1253–1262
21. Knudsen, S.: *A Biologist’s Guide to Analysis of DNA Microarray Data*. Wiley Pub. (2002)
22. Baldi, P., Hatfield, G.W.: *DNA Microarrays and Gene Expression*. Cambridge University Press (2002)