Evolutionary insights from suffix array-based genome sequence analysis

Anindya Poddar[1], Nagasuma Chandra[1,2], Madhavi Ganapathiraju[1,3], K Sekar[1,2], Judith Klein-Seetharaman[3], Raj Reddy[3] and N Balakrishnan[1,3*]

[1]*Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore 560 012, India*
[2]*Bioinformatics Centre, Indian Institute of Science, Bangalore 560 012, India*
[3]*Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA*

**Corresponding author (Fax, 91-80-22933438; E-mail, balki@serc.iisc.ernet.in)*

Gene and protein sequence analyses, central components of studies in modern biology are easily amenable to string matching and pattern recognition algorithms. The growing need of analysing whole genome sequences more efficiently and thoroughly, has led to the emergence of new computational methods. Suffix trees and suffix arrays are data structures, well known in many other areas and are highly suited for sequence analysis too. Here we report an improvement to the design of construction of suffix arrays. Enhancement in versatility and scalability, enabled by this approach, is demonstrated through the use of real-life examples.

The scalability of the algorithm to whole genomes renders it suitable to address many biologically interesting problems. One example is the evolutionary insight gained by analysing unigrams, bi-grams and higher n-grams, indicating that the genetic code has a direct influence on the overall composition of the genome. Further, different proteomes have been analysed for the coverage of the possible peptide space, which indicate that as much as a quarter of the total space at the tetra-peptide level is left un-sampled in prokaryotic organisms, although almost all tri-peptides can be seen in one protein or another in a proteome. Besides, distinct patterns begin to emerge for the counts of particular tetra and higher peptides, indicative of a 'meaning' for tetra and higher n-grams.

The toolkit has also been used to demonstrate the usefulness of identifying repeats in whole proteomes efficiently. As an example, 16 members of one COG, coded by the genome of *Mycobacterium tuberculosis* H37Rv have been found to contain a repeating sequence of 300 amino acids.

[Poddar A, Chandra N, Ganapathiraju M, Sekar K, Klein-Seetharaman J, Reddy R and Balakrishnan N 2007 Evolutionary insights from suffix array based genome sequence analysis; *J.Biosci.* **33** 871–881]

1. Introduction

The current practice in computational biology and bioinformatics involves an essential and a crucial component of sequence analysis upon which several further investigations are carried out and higher-level knowledge is acquired. Genome sequences, both as nucleic acids or as their translated proteomes, are essentially sequences of strings, and are therefore routinely analysed by various string matching and searching algorithms. With recent advances in sequencing technology, several genomes have been sequenced in the last few years, leading to an unprecedented growth of the sequence databases. Availability of information of such large magnitude has given rise to a new tide in biology research, much of it dependent fundamentally on computational sequence analysis. Although several algorithms have emerged in the recent past to carry out such analysis, there is still a high potential for improving the

Keywords. Biological language modelling toolkit (BLMT); genome sequence analysis; n-grams; pattern matching; suffix arrays; suffix trees; short peptide sequences genetic code bias

efficacy of computational sequence analysis, both in terms of speed as well as in terms of flexibility and adaptability of the tools to address different biological issues. This is specially true for addressing evolution related questions. In particular, no analysis has been reported so far on what proportion of the total peptides of different lengths that are in principle possible to occur, are actually observed in different organisms and whether they differ among different life forms. Large scale cross comparisons of whole genomes and the proteomes they code for, are required to study this issue, which requires efficient algorithms.

Genome sequences, essentially being linear sequences of symbols indicating various genes within it can be easily viewed as a string consisting of a set of biologically meaningful sub-strings. Due to the large size of the genome data, efficient searching for sub-strings poses several challenges. String matching and pattern recognition have been well studied in other fields such as data compression, information retrieval, word processing and language modelling. Learning from the successes in these areas, we understand that appropriate representation of data holds the key for developing efficient sequence analysis algorithms. Suffix trees and suffix arrays have been shown to be efficient data structures that enable fast comparison of sub-strings, through methods such as the n-gram analysis. Numerous applications using these have emerged for genome and protein sequence processing, beginning with the introduction of generalized suffix trees for biological sequence analysis (Bieganski *et al* 1993). N-gram statistics have been presented in (Ganapathiraju *et al* 2002; Klein-Seetharaman *et al* 2002) and model based comparisons of n-grams indicating long distance correlations in amino acids are presented in (Beuhler and Ungar 2001). Pattern matching algorithms specifically designed for genome and protein sequences have been developed such as q-gram based database searching using suffix arrays (Burkhardt *et al* 1999), whole genome alignment using suffix trees (Delcher *et al* 1999), sequence clustering (Malde *et al* 2003), regular expression matching (Sivaraman *et al* 2003), computation of maximal repeats in whole genomes (Irving and Love 2001), efficient discovery of proximity patterns (Arimura *et al* 2001), protein family modelling using probabilistic suffix trees (Bejerano and Yona, 2001), and binary search trees for indexing DNA with suffix trees (Hunt *et al* 2000) and with suffix arrays (Irving and Love 2001). Algorithms presented in the areas of natural language processing such as suffix arrays for statistical language modelling (Rosenfeld 1997), for Yule-value computations and for computing term frequency and inverse document frequency in the domain of information retrieval (Yamamoto and Church 2001) are also applicable by analogy for genome sequence analysis (Sivaraman *et al* 2003; Ganapathiraju *et al* 2002, 2004 a,b,c). Although many of these are currently being used for biological research, each method has its own limitations of which many of them pertain mainly to the time involved in pre-processing the data, warranting development of newer methods to overcome such limitations.

A recent review summarises the problems and complexity involved and the taxonomy of methods that are available to construct suffix arrays (Puglisi *et al* 2007), highlighting the need to overcome this problem for efficient use of the technique.

The biological language modelling toolkit (BLMT) developed at Carnegie Mellon University, based on suffix arrays, is one of its kind, in that it makes it readily available for the biological and bioinformatics community to use the tools for sequence analysis through a web-interface (Manoharan *et al* 2006), and also makes the toolkit available in Open Source (Ganapathiraju *et al* 2004a,b,c), for the computational community to develop new algorithms or to improvise existing algorithms. In this paper, we present an augmentation to the toolkit in terms of scalable linear time construction of the suffix array data structure, through a linear time construction of suffix tree (Ukkonen 1995). This extends the applicability of the BLMT to larger data sizes than previously supported. Further, significant biological observations made possible by this efficient preprocessing are also presented in the paper.

2. Methods

2.1 *Suffix array and longest common prefix values construction*

A genome or proteome sequence can be preprocessed in the form of suffix tree or suffix array in such a fashion that subsequences forming specific patterns or repeats can be accessed efficiently (figures 1, 2A) (Ganapathiraju *et al* 2004a,b,c). The bottleneck of suffix tree is that for an alphabet of size Σ it consumes $O(N|\Sigma|)$ space, where N is the length of the sequence. For proteomes that have an alphabet size Σ of 20, this imposes a restriction for storage in main memory. Suffix arrays that require only $O(N)$ space prove to be a better choice for large proteomes. The suffix array data structure is an array of N integers indicating the positions of all the suffixes in lexicographical order for a string of N characters. Linear time construction of suffix array is achieved from linear time construction of suffix tree through lexical depth-first traversal by Ukkonen algorithm (Ukkonen 1995). The suffix array and suffix tree are constructed for each of the genomes separately and are stored on hard disk. Although all the applications discussed in this paper are built over the suffix arrays, the suffix trees are also stored for possible future applications specific to this data structure.

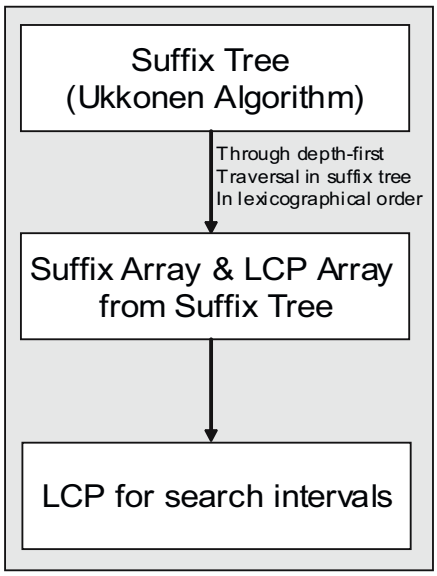

Figure 1. Stages in data preprocessing: A suffix tree is first constructed from the data, from which suffix array and the LCP array are constructed. LCP search intervals are then determined for faster sequence searching.

This construction of the data structure for individual genomes is a one time computation and its resulting array is readily available for all further computations.

For efficiently counting the missing n-grams, regular expressions and motifs in a proteome database, the longest common prefix (LCP) array is used over the suffix array. It is an array of integers indicating the length of the longest common prefix between two consecutive suffixes in the suffix array. LCP array is constructed in linear time using (Kasai *et al* 2001). LCP search intervals, which further enhance the speed of string searches in suffix arrays, are also constructed (figure 2B). A binary tree of LCP search intervals is constructed as given in (Abouelhoda *et al* 2002). It consists of a binary tree, wherein all the possible LCP values occur as the leaf nodes and the search intervals are represented by the internal nodes. If N is a power of 2, then altogether there will be $(2N–1)$ LCP values (Gusfield 1997).

Also to reduce the time and space used for single pattern matching, the required suffix array and LCP array elements are selectively chosen from the hard disk (Burkhardt *et al* 1999).

3. Results and discussion

3.1 *The toolkit: Improvement in performance and scalability*

A suffix array for a small genome sequence of 1.6 MB, built by first constructing suffix trees, using the Ukonnen algorithm, requires 15.26 s, in contrast to the suffix array built using the inplace-binary sort with a 3-character radix (CMU BLMT) that required 283.2 s. The small cost in additional storage space required for the suffix tree approach does not pose a significant problem, given the present advances in hardware technology. On the other hand, reducing the pre-processing time offers a significant advantage for the application by substantially alleviating the drawback of the time required in the initialization phase due to the use of suffix arrays constructed using in place-binary sort with a 3-character radix method. Figure 3 demonstrates

Table 1. Demonstration of Scalability for various string operations for different file sizes: Machine: Sun-Fire-880 (UltraSPARC-III); CPU Frequency: 750 MHz; Memory: 32768 MB

Database	Size	Storage	ST Creation time	Pre-processing	Motif (AAAA) Searching CPU Time	Missing N = 4	Missing N = 5	Present N=15	Present N=100	Present N=200
Bacteria: *Mycobacterium tuberculosis* H37Rv	1.6	29.04	10.25	15.26	0.1	0.29	1.23	0.39	1.06	1.63
Bacteria: *Streptomyces avermitilis*	3.19	57.47	24.27	34.54	0.14	1.53	39.5	0.77	2.12	3.21
Eukaryote: *C.elegans*	9.59	220.87	167.51	235.31	0.21	3.04	44.51	3.61	8.63	13.41
NR (Portion of)	55	984.02	2181.57	2613.61	1.56	9.45	55.05	25.25	62.6	93.94

All data sizes shown are in MB, and the times are CPU times in s. The numbers are also shown plotted in figure 4 to indicate the linear relation of time to compute with respect to the size of the data.

the improvement in pre-processing efficiency, as compared to the CMU toolkit.

The improvement in efficiency has been observed in a near-linear fashion for larger genomes as well. Table 1 details the storage requirements and the pre-processing times for 4 different datasets, ranging from 1.6 to about 55 MB in size. The time requirements are shown in figure 4, to demonstrate the linear dependence in time of computation to the size of the database. The efficient implementation with which suffix arrays can be constructed for large genomes and also the entire non-redundant sequence database (NR), renders it practical to carry out many of global analyses of the protein and DNA sequences.

3.2 *Performance comparison with GCG software and Boyer-Moore algorithm*

Primarily, we found for unigram count of *Mycobacterium tuberculosis* H37Rv that, while GCG software takes 49.20 s in SGI IP32 processor with 300 MHz CPU frequency, the suffix array method takes only 0.51 s on the same machine.

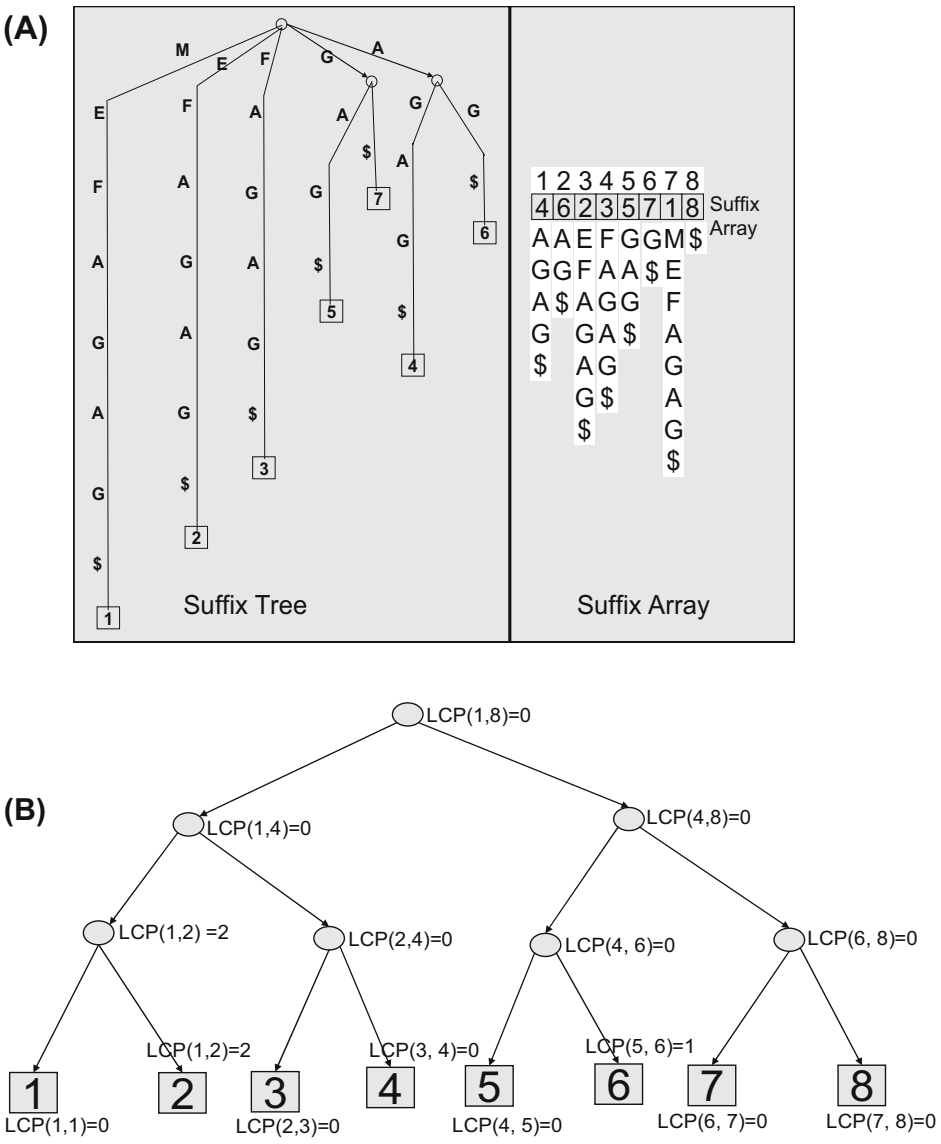

Figure 2. **(A)** Example of a suffix tree and suffix array for the string "MEFAGAG". The string is concatenated with the character "$" in both Suffix tree and array constructions. The left frame shows suffix tree for the given string. Traversing down each branch of the tree gives rise to a suffix. In the right frame, the suffix array is shown for the same string. Top row shows the position index in the suffix, second row shows the indices of suffixes in lexicographical order. The corresponding suffixes are shown hanging vertically from each position. **(B)** LCP search intervals for the example suffix array shown in **(A)**.

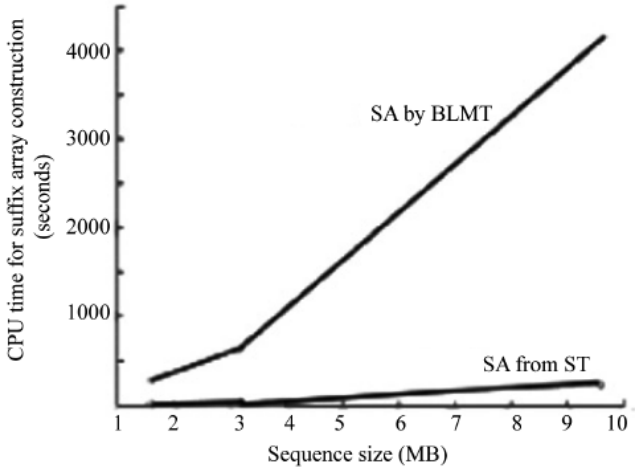

Figure 3. Demonstration of linear time requirement for computation of suffix array using suffix trees. As can be seen the suffix array construction time is reduced significantly by constructing suffix tree as an intermediate step. Even at 10 MB datasize, the difference in time ~3500 seconds or close to 1 h. For larger data sizes, such as human proteome, non-redundant protein database, etc the reduction in computational time makes a significant difference.

For the same sequence, when single pattern matching is concerned, the GCG package takes 52.51 s to search for a 4-gram motif, while using suffix array technique we compute the same at 1.28 s, again on the same platform. The results for three genomes are given in tables 2A and 2B and also illustrated in figure 5. While comparing with Boyer-Moore algorithm, we found that the suffix array technique outperforms the Boyer-Moore algorithm at larger sequence sizes.

3.3 *Example analysis to demonstrate the usefulness of the augmented toolkit leading to new biological insights*

The improvement to the design of a suffix array based genome analysis toolkit, reported here, has significantly reduced the pre-processing time. The suffix array construction algorithm has also been optimized for storage capacity and preprocessing and search times, with the augmentation of the LCP search-intervals array. The scalability of the algorithm renders it suitable to address many biologically interesting problems. To demonstrate this, the toolkit has been applied to a few examples chosen (i) to validate its functionality and performance, and (ii) demonstrate its usefulness in carrying out various kinds of analysis of large scale genomic data easily and efficiently. The performance of the toolkit has been compared to one of the widely used methods, where appropriate. Data structures, their representation and the design of the algorithms used in this toolkit, also enable newer lines of investigations, that have not been carried out earlier. Such analysis has in fact led to interesting biological observations and evolutionary insights, which are described below.

Table 2A. Time for unigram count and single pattern matching using GCG and suffix array techniques: M/c: SGI IP-32, CPU Frequency: 300 MHz (time in seconds)

		Mycobacterium tuberculosis H37Rv	*Streptomyces avermitilis*	*C.elegans*
Searching type	Softwares	1.60 MB	3.19 MB	9.59 MB
Unigram count	Suffix array	0.51	1.22	4.3
	GCG	49.2	96.98	280.48
Single pattern matching (pattern = "AAAA")	Suffix array	1.28	1.47	1.91
	GCG	52.51	140.44	294.45

The order of time of computation using suffix arrays has been found to be the same for patters in any lexicographical position, that is for 'AAAA' or 'WPLK', and hence the lower time achieved is not due to the specific choice of the 4-gram.

Table 2B. Time comparison between suffix array and Boyer Moore technique for single pattern matching M/c: Sun Blade-1000, CPU Frequency: 900MHz. (Time in seconds)

Searching type	Algorithms	*Mycobacterium tuberculosis H37Rv*	*Streptomyces avermitilis*	*C.elegans*
		1.60 MB	3.19 MB	9.59 MB
Single pattern matching (Pattern = "AAAA")	Suffix Array	0.1	0.14	0.2
	Boyer-Moore	0.07	0.14	0.53

Note that the hardware platform of computation given in tables A and B are different.

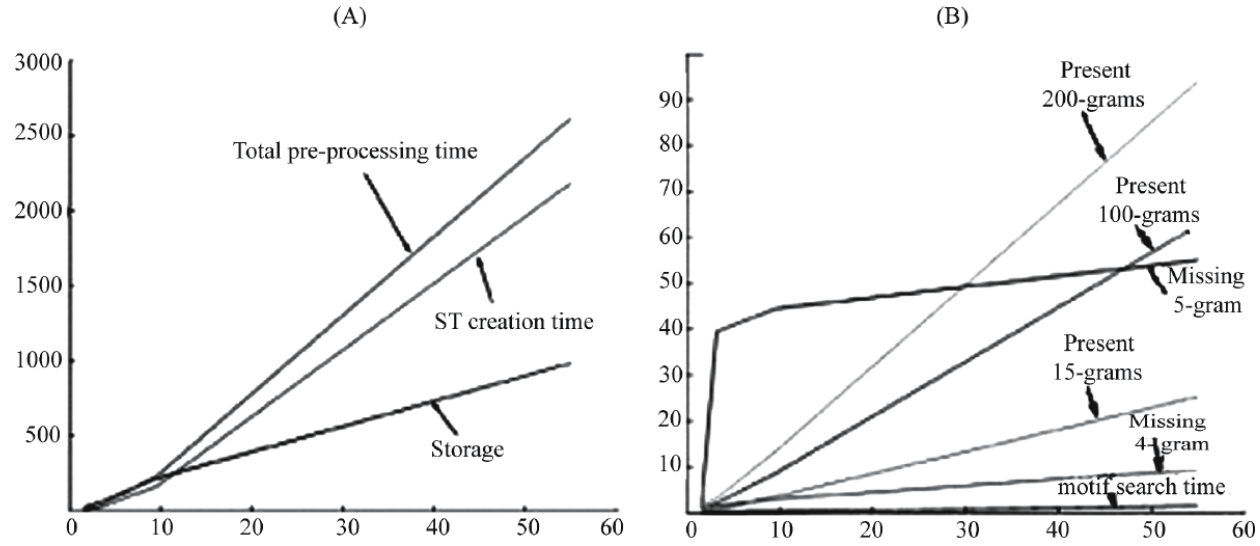

Figure 4. Demonstration of linear relation of storage and computation time with respect to size of data: The data in table 1 are presented here in the plots for a clear demonstration of the linear relation. **(A)** Total time in seconds for preprocessing the data structures (suffix arrays, LCP array, LCP interval array and the rank array), of which a large component in the suffix tree creation time, the storage requirement for all of the data in MB are shown. The plot corresponds to columns 1 to 5 of table 1. **(B)** For the same data, the time in seconds for computation of 5-grams, 15-grams, 100-grams and 200-grams present in the data, and 4-grams missing in the data, and the time to search for a specific 4-grams are shown. The plot corresponds to columns 6-11 with respect to data in columns 1-2 of table 1.

3.4 *Redundancy in the genetic code dictates overall genome compositions*

Analysis of the unigram distributions of various genomes indicates that amino acids which are coded by multiple codons occur more frequently than those for which fewer codons exist. Even among those amino acids that are coded by only two codons, in the standard genetic code, the occurrences of cysteine, tryptophan and methionine were fewer (figure 6) and could be linked to the fact that their codons, when changed in the third position lead to *stop* or *start* codons, which would be detrimental to the protein and therefore not easily preferred during evolution. This also suggests that these amino acids, in particular, the cysteine and the tryptophan, are not incorporated into the proteins unless they play specific roles. The genetic code is thus optimally designed to reserve the sparingly used triplet codes to be *near* to each other, and farther from other frequently used codes, thereby avoiding accidental point mutations resulting in these drastically affecting codons. The 2-gram and higher n-gram segments or 'phrases' containing these amino acids, where present, indicate a significance either for protein structure or function than other segments of the same size.

This observation common to various life forms analyzed here, is consistent with the theory of evolution being random (for e.g. Caporale 1999), because, the higher the chances for a particular amino acid to be coded, the higher is its usage in the genome. Diversity between genomes is brought about by deviations from the standard code itself. This is illustrated by the higher percentage of tryptophans and a significantly lower percentage of arginines in the metazoan mitochondrial genome, consistent with the alterations in its genetic code, which indicates that two of the six codons for arginine in the standard genetic code are converted to termination

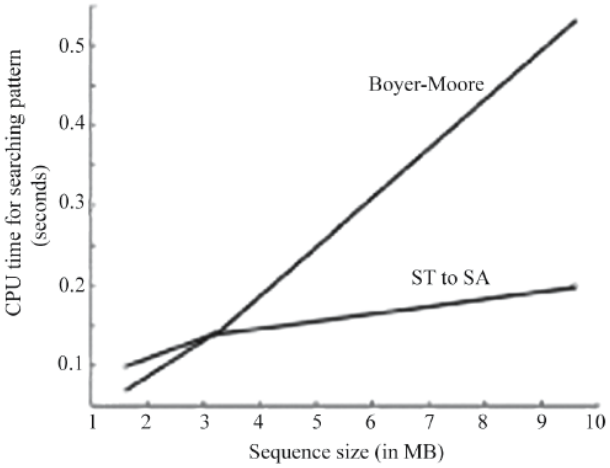

Figure 5. Time for pattern searching with this toolkit versus Boyer-Moore algorithm. While at very small data sizes Boyer-Moore algorithm outperforms suffix arrays toolkit, the latter very quickly over takes the Boyer-Moore algorithm when data size becomes larger. Once one occurrence of required pattern is found, locating all occurrences of the same pattern requires minimal time with suffix arrays and is thus suitable for specific applications requiring such searches.

codons in this organism, whereas a termination codon in the standard code is converted to a tryptophan (*http://www.ncbi.nlm.nih.gov/Taxonomy/*).

Yet, within this overall framework, significant differences between preferences of amino acids and the various combinations in terms of bigrams, trigrams and higher peptides vary from organism to organism. Ganapathiraju and coworkers (2002) have reported, that a simple Markovian unigram model distinguishes different organisms, suggesting that, different organisms use different vocabulary, perhaps optimized for their survival. Our findings further support this argument, for reasons described below. The unigram counts of the genome of *M. tuberculosis* (figure 6A), shows a higher percentage of arginines, alanines and prolines, all coded by combinations of guanine and cytosine. This genome is known to be GC rich (Cole *et al* 1998), despite

Table 3. Statistics of different types of N-gram counts in several proteomes. Corresponding file sizes (byte) are indicated

Proteomes	Size (B)	$N = 2$		$N = 3$		$N = 4$	
		Present	Missing	Present	Missing	Present	Missing
Archaea							
Aeropyrum pernix	712,538	400	0	7980	20	110838	49162
Methanothermobacter thermautotrophicus	578,192	400	0	7970	30	111930	48070
Sulfolobus tokodaii	837,491	400	0	7965	35	117398	42602
Methanocaldococcus jannaschii	530,204	405	0	7948	68	103556	56468
Archaeoglobus_ fulgidus	734114	400	0	7987	13	118726	41274
Bacteria							
Bacillus cereus ATCC 14579	1,598,624	400	0	7996	4	140824	19176
Aquifex aeolicus VF5	526,311	401	0	7953	49	104670	55333
Agrobacterium tumefaciens str. C58 (Cereon)	921,895	400	0	7992	8	126339	33661
Mycobacterium tuberculosis H37Rv	1,650,780	400	0	7998	2	130336	29664
Mycobacterium tuberculosis CDC1551	1,763,089	421	0	8137	1	131291	28970
Mycobacterium leprae	693,138	400	0	7972	28	109953	50047
Eukaryota							
Caenorhabditis elegans	10,051,500	407	0	8014	0	159469	552
Drosophila melanogaster	7,030,281	401	0	8002	0	159021	982
Arabidopsis thaliana – CHR 1	3,946,998	404	0	8007	0	155189	4821
Arabidopsis thaliana – CHR 2	2,383,404	400	0	8000	0	150683	9317
Arabidopsis thaliana – CHR 3	2,979,379	400	0	8000	0	153052	6948
Arabidopsis thaliana – CHR 4	2,318,251	400	0	8000	0	150072	9928
Arabidopsis thaliana – CHR 5	3,441,836	400	0	8000	0	154331	5669

The columns show number of distinct n-grams present and absent for values of $n = 2$, 3 and 4. For NR database, although all of the possible 4-grams are found, the number missing 5-grams is found to be 307,303.

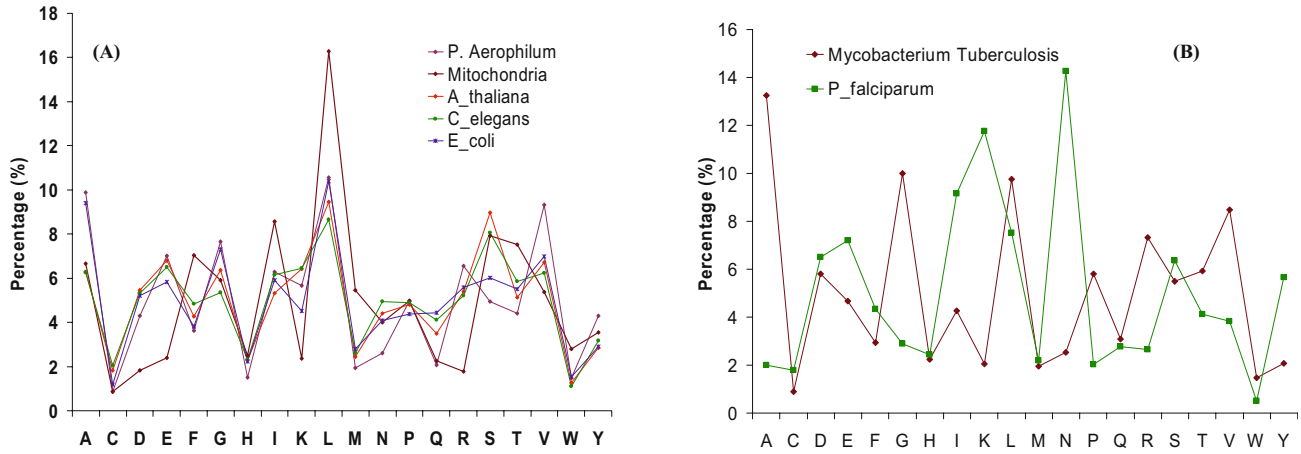

Figure 6. **(A)** Unigram distribution (percentage) in the proteomes of A*erophilum,* Mitochondria*, A. thaliana, C. elegans* and *E. coli*. **(B)** Unigram distributions in the genomes of *M. tuberculosis* and *Plasmodium falciparum*.

conforming to the standard genetic code. Interestingly, the genome codes for a number of PE and PPE proteins, unique to mycobacteria that are rich in alanines, prolines, glycines and arginines. The unigrams of *P. falciparum* (figure 6B), on the other hand shows an unusual richness in asparagines and lysines, matching with the known AT richness of this genome (Gardner *et al* 2002). This type of analysis would enable classification and grouping of organisms based on similarities in the unigram counts and help in exploring if unigram preferences are conserved across different species of a given genus. The results obtained here also lay a foundation to explore the biological significances of significant changes in individual genomes.

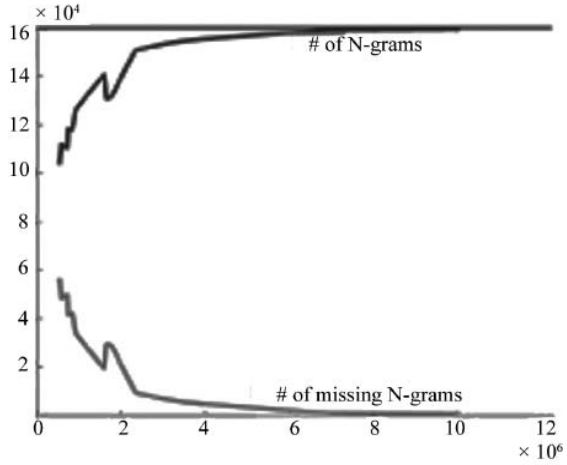

Figure 7. Variation in number of 4-grams and missing 4-grams with increase in data size. For 1-2 sizes of *n*, all n-grams are present in even small proteomes. For n=3, some n-grams are missing in bacterial genomes, although the available number of non-overlapping 3-grams is several times larger than the number of distinct 3-grams. The figure shows the number of 4-grams present and missing in genomes, for increasing sizes of the genomes. Even at 10 MB genome size, about 500 4-grams are not present in thegenome.

3.5 *Analysis of the coverage of peptide space in different life forms reveals higher 'meaning' for longer N-grams*

There are no reports in the literature so far which indicate how efficiently evolution has utilized the available peptides of different lengths or in other words the peptide space, in proteomes of different organisms. It is of interest to determine if some combinations are preferred over others, which might throw some light onto the functional roles of individual amino acids in different contexts in different proteins and what constraints they may pose during evolution. Although such questions can be answered easily by relevant single-molecule experiments or analysis, identifying patterns of occurrences of smaller peptide units can serve as a stepping stone. An analysis of the bigram, trigram and higher peptides present in each genome was therefore carried out, in an effort to explore if genomes have evolved to make use of amino acid combinations efficiently. The toolkit was applied to archaeal, bacterial and eukaryal genomes to study the present and missing n-grams for several values of n. The results (shown in table 3) indicate that all genomes contain all 20 amino acids, and also contain all possible (400) bigrams arising out of these 20 amino acids, irrespective of their unigram distributions. Among the trigrams, the eukaryal genomes had all the 8000 combinations, but the bacterial genomes had a few (about 30 on average) combinations missing in them. This is despite the fact that, for a genome of about 4000 proteins summing to 4 MB, there are at least 1,275,333 non-overlapping possibilities for a given trigram to occur in the genome. When extended to 4-grams, it was

Table 4. Absolute repeats in Protein Sequences (Protein identification numbers, known annotations and positions of repeating elements in the *M. tuberculosis* genome). The repeating sequence in all the three cases is also shown

GI number	Rv number	Protein name	Position of repeat
RWGVESICTQLTELGVPIAPSTYYDHINREPSRRELRDGELKEHISRVHAANYGVYGARKVWLTLNREGIE VARCTVERLMTKLGLSGTTRGKARRTTIADPATARPADLVQRRFGPPAPNRLWVADLTYVSTWAGFAYV AFVTDAYARRILGWRVASTMATSMVLDAIEQAIWTRQQEGVLDLKDVIHHTDRGSQYTSIRFSERLAEAGI QPSVGAVGSSYDNALAETINGLYKTELIKPGKPWRSIEDVELATARWVDWFNHRRLYQYCGDVPPVELE AAYYAQRQRPAAG			
gi|2911036|emb|CAA17525.1|	796	hypothetical protein	20
gi|2894236|emb|CAA17098.1|	3326	hypothetical protein	20
gi|3261683|emb|CAB06167.1|	2355	hypothetical protein	20
gi|3261653|emb|CAB03675.1|	2814c	hypothetical protein	20
gi|2827595|emb|CAA16650.1|	3185	hypothetical protein	20
gi|3261821|emb|CAB10723.1|	2106	hypothetical protein	20
gi|3261611|emb|CAB00998.1|	2279	hypothetical protein	20
gi|2827597|emb|CAA16652.1|	3187	hypothetical protein	20
gi|2104398|emb|CAB08699.1|	3475	hypothetical protein	54
gi|3242294|emb|CAA17494.1|	2167c	hypothetical protein	54
gi|2791519|emb|CAA16056.1|	2479c	hypothetical protein	54
gi|3242268|emb|CAB02367.1|	2649	hypothetical protein	36
gi|2131024|emb|CAB09338.1|	1764	hypothetical protein	2
gi|1621251|emb|CAB02647.1	1369c	hypothetical protein	2
gi|2131021|emb|CAB09336.1|	1756c	hypothetical protein	2
gi|2661659|emb|CAA15765.1|	3380c	hypothetical protein	2
KKGNIPVGYYKDEKKTAETFRTINGVRYAIPGDYAQVEEDGTVTMLGRGSVSINSGGEKVYPEEVEAALK GHPDVFDALVVGVPDPRYGQQVAAVVQARPGCRPSLAELDSFVRSEIAGYKVPRSLWFVDEVKRSPAG KPDYRWAKEQTEARPADDVHAGHVTSG			
gi|2924450|emb|CAA17750.1|		fadD18	52
gi|2924452|emb|CAA17752.1|		fadD19	382
MRSKIPDLQRALEGRFDDHHALMCRLHLAHLDQLDAMIGALDEQIEQLMHPFCARRELIASIPGIGVGASA TVISEIGADPAAWFPSAEHLASWVRLCPGNHESAGKRHHGARRTGNQHLQPVLVECAWAAVRTDGYLR EYYRRQVRKFGGFRSPAANKKAI			
gi|1666161|emb|CAB03772.1|	2424c	hypothetical protein	113
gi|2911097|emb|CAA17481.1|	2177c	hypothetical protein	1

observed that there were a number of combinations (about 40,000 on average) that are not present in archaea or bacteria (figure 7). Even in the eukaryal genomes with sizes many times larger, there were about 500-900 4-grams missing (table 3). These peptides were not the same in all the genomes and differed heavily from genome to genome, strongly suggesting that tri and higher-peptides begin to indicate 'meaning'. A pattern of conserved phrases containing tri- and higher peptides in either proteins belonging to a family or in genomes of closely related organisms could therefore lead to deriving appropriate fingerprints. The non-redundant protein sequence database was also analyzed to identify if there were any tri- and tetra-ngrams not present in any life form whose sequences are known so far. The results given in table 3 show clearly that all 8000 tri-grams were present in one organism or the other whereas 15 tetra-grams were not present in any of them. These were predominantly peptides containing either cysteine or tryptophan or both (consistent with their low unigram counts), suggesting that such combinations arising from these residues exert strong negative selection pressures during evolution.

It is well known that proteins survive the strongest evolutionary constraints since they need to retain their respective functional roles and therefore tolerate only changes that do not alter their function significantly. Their conservation is found to be much higher at the structural level than at the sequence level as judged by the numerous examples in literature. This is because a given structure (and perhaps function too) can be achieved through different sequence sets, whose relationships are not obvious by sequence comparisons alone. However there may be preferences for the usage of specific tri, tetra- and higher peptide segments in individual organisms, which if accounted for, would help in identifying sequence similarities better. Ganapathiraju *et al* (2002) have

shown earlier that simple Markovian unigram models are characteristic of organisms and can be used for distinguishing them. The analysis presented here further strengthens the argument and shows that, tri- and higher peptides emerge to have specific 'meaning' and can be perhaps be considered to be equivalent to 'words' in natural language.

The differences in compositions of proteomes, although all made by the same 20 amino acids are sufficient to result in characteristic features of that genome. This also illustrates one of the fundamental concepts put forth by Darwin, which was to recognize that the key to understanding the biological world lay in the minor variations

3.6 *Analysis of repeats: 16 members of one COG identified in the genome of* Mycobacterium tuberculosis

Analysis of repeats within the *M. tuberculosis* H37Rv genome shows that 16 different ORFs, about 300 amino acids in length, distributed across the genome are identical (table 4). These proteins are believed to be transposases, required for the transposition of the insertion elements IS6110. Insertion elements have been correlated with pathogenicity (Brosch *et al* 2001) and also have been reported to be useful genetic markers for identifying isolates of the *M. tuberculosis* complex and for distinguishing between different strains (Fang *et al* 1999).

Apart from this, the analysis also identifies internal repeats within the same ORFs. Some examples are repeats of 200 to 250 amino acid stretches in a PE_PGRS protein (Rv3507), a PPE protein (Rv3343c) and in a probable polyketide synthase pks12. It is interesting to note that no such internal repeats are found in a related genome *Mycobacterium leprae*. However, even in a minimalist genome such as the *M. leprae* (Cole *et al* 2001), repeating domains of at least 200 residues were found in 3 pairs of proteins corresponding to a L-asparagine permease and an aromatic amino acid transport protein; in polyketide synthases and in a hypothetical protein that could be identified as a putative myo-inositol-1-phosphate synthase.

Analysis of the repeats with a genome also helps in identifying domain rearrangements. Several such changes in domain arrangements where an identical 200 amino acid stretch was inserted in different places in different proteins were identified in the *M. tuberculosis* genome. For example, the locations of the N-terminal domain of Rv0058, a segment of the replicative DNA helices were found to be present in different locations in different proteins within the same genome. Identification of repeats in different organisms can also serve as a first guide to understand horizontal gene transfer.

4. Conclusion

In this paper, we present an implementation of suffix arrays for genome sequence analysis, that helps in significantly reducing pre-processing time, making the algorithm readily usable for large scale analysis. The suffix array construction algorithm has also been optimized for storage capacity and preprocessing and search times, with the augmentation of the LCP search-intervals array. The software developed using this approach has been ported to almost all of the well-known processors, and is made available in Open Source. This has made the tool far more useful, through scalability to larger data sizes, and through extension of the foundation data structures to include LCP search interval array that extends support to more applications over the toolkit.

Further, the ability to carry out large scale genome analysis and cross-comparisons across genomes leads to new insights in biology, most prominent of them being evolutionary processes. A study of the unigram distributions of various genomes reveals that redundancy in the genetic code dictates the overall composition of a given genome. Yet, within the overall framework, significant preferences for particular combinations of amino acids become apparent. This aspect becomes even clearer from the analysis of the coverage of peptide space in different life forms, which reveals that tri- and higher peptides emerge to show 'meaning', which can perhaps be considered to be equivalent towards meaning in natural language, thus strengthening the previous argument that different organisms use different vocabulary.

The toolkit has also made it possible to detect transposases in genomes, and also internal repeats of sequences internal to specific ORFs, via n-gram analyses. Genome sequence analysis through n-grams has many other potential applications, specifically in the study of conserved peptide sequences within protein families. The enhanced toolkit has been shown to be scalable to the entire non-redundant sequence data base and thus, potentially forms a software foundation over which other applications and analysis tools may be built. The availability of the source code of the toolkit leads to other many useful applications, of which detection of internal repeats is shown here as an example.

Acknowledgements

The authors thank Prof. M Vijayan for useful discussions. The financial support from the Department of Biotechnology Computational Genomics Initiative at the Indian Institute of Science (IISc) is gratefully acknowledged. Use of facilities at the Super Computer Education and Research Center and the Bioinformatics Centre of IISc (the latter supported by Department of Biotechnology), are also gratefully acknowledged.

References

Abouelhoda M I, Kurtz S and Ohlebusch E 2002 The enhanced suffix array and its applications to genome analysis; *Proceedings of the Second Workshop on Algorithms in Bioinformatics*, September 17–21 (Springer-Verlag) pp 449–463

Arimura J, Asaka H, Sakamoto H, Arikawa S 2001 *Efficient discovery of proximity patterns using suffix arrays;* July 1–4, Jerusalem, Israel

Bejerano G and Yona G 2001 Variations on probabilistic suffix trees: statistical modeling and prediction of protein families; *Bioinformatics* **17** 23–43

Beuhler E C and Ungar L H 2001 Maximum entropy methods for biological sequence modeling; in *Workshop on Data Mining in Bioinformatics 2001* (BIOKDD 2001) pp 60–64

Bieganski P, Riedl J, Carlis J Retzel E F 1994 Generalized Suffix Trees for Biological Sequence Data. 1994 *System Sciences* V: Biotechnology Computing; in *Proceedings of the Twenty-Seventh Hawaii International Conference*, University of Minnesota, vol 5, pp 35–44

Brosch R, Pym A S, Gordon S V and Cole S T 2001 The evolution of mycobacterial pathogenicity: clues from comparative genomics; *Trends Microbiol.* **9** 452–458

Burkhardt S, Crauser A, Ferragina P, Lenhof H-P, Rivals E, *et al* 1999 q-gram based database searching using a suffix array (QUASAR); in *RECOMB, Annual Conference on Research in Computational Molecular Biology*, *Proceedings*, Lyon, France, pp 77–83

Caporale L H 1999 Chance favors the prepared genome; *Ann N. Y. Acad. Sci.* **870** 1–21

Cole S T, Brosch R, Parkhill J, Garnier T, Churcher C *et al* 1998 Deciphering the biology of *Mycobacterium tuberculosis* from the complete genome sequence; *Nature (London)* **393** 537–544

Cole S T, Eiglmeier K, Parkhill J, James K D, Thomson N R *et al* 2001 Massive gene decay in the leprosy bacillus; *Nature (London)* **409** 1007–1011

Delcher A L, Kasif S, Fleischmann R D, Peterson J, White O *et al* 1999 Alignment of whole genomes; *Nucleic Acids Res.* **27** 2369–2376

Fang Z, Doig C, Morrison N, Watt B and Forbes K J 1999 Characterization of IS1547, a new member of the IS900 family in the *Mycobacterium tuberculosis* complex, and its association with IS6110; *J. Bacteriol.* **181** 1021–1024

Ganapathiraju M, Klein-Seetharaman J, Balakrishnan N and Reddy R 2004a Characterization of protein secondary structure using latent semantic analysis. *IEEE Signal Processing magazine*, May 2004, issue 15, 78–87

Ganapathiraju M, Manoharan V and Klein-Seetharaman J 2004b BLMT: Statistical Sequence Analysis using N-grams; *J. Appl. Bioinformatics* **3** 193–200

Ganapathiraju M, Weisser D, Klein-Seetharaman J and Reddy R 2004c Yule value tables from protein datasets of different categories: emphasis on trasnmembrane proteins; *Proc. SCI2004*, Florida, USA

Ganapathiraju M, Weisser D, Rosenfeld R, Carbonell J and Reddy R *et al* 2002 Comparative n-gram analysis of whole-genome sequences; *HLT'02: Human Language Technologies Conference*, San Diego, March, 2002. San Diego, USA

Gardner M J, Hall N, Fung E, White O, Berriman M *et al* 2002 Genome sequence of the human malaria parasite *Plasmodium falciparum*; *Nature (London)* **419** 498–511

Gusfield D 1997 *Algorithms on strings, trees and sequences* (Cambridge University Press)

Hunt E, Irving R W and Atkinson M 2000 Persistent Suffix Trees and Suffix Binary Search Trees as DNA Sequence Indexes. Glasgow: Department of Computing Science, University of Glasgow. *TR-2000-63*

Irving R W and Love L 2001 suffix binary search trees and suffix arrays. Dept of Computing Science, University of Glasgow. *TR-2001-82*

Kasai T, Lee G, Arimura H, Arikawa S, Park K. 2001 Linear-Time Longest-Common-Prefix computation in Suffix Arrays and Its applications; *Lecture Notes in Computer Science, Combinatorial Pattern Matching: 12th Annual Symposium, CPM 2001*, July 1-4, Israel, Proceedings, 181–192

Klein-Seetharaman J, Ganapathiraju M, Rosenfeld R, Carbonell J and Reddy R 2002 Rare and frequent amino acid n-grams in whole-genome protein sequences; 2002; *RECOMB'02: The Sixth Annual International Conference on Research in Computational Molecular Biology*, Washington DC, USA

Malde K, Coward E and Jonassen I 2003 Fast sequence clustering using a suffix array algorithm; *Bioinformatics* **19** 1221–1226

Manoharan V, Ganapathiraju M and Klein-Seetharaman J 2006 Ambient Intelligence Everyday Life; in *Lecture notes in computer science* (eds) Y Cai, J Abascal, (Springer) (in press)

Puglisi, S J, Smyth,W F and Turpin, A H 2007 A taxonomy of suffix array construction algorithms; *ACM Comput. Surv.* 39, 2, Article 4, June

Rosenfeld R 1997 CMU *Cambridge statistical language modeling toolkit* (Proceedings ESCA Eurospeech)

Sivaraman B, Ganapathiraju M, Klein-Seetharaman J, Balakrishnan N and Reddy R 2003 *Extensions to biological langauge modelling toolkit (BLMT)*; Pittsburgh, USA

Yamamoto M and Church KW 2001 Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus; *Comput. Linguist.* **27** 1–30

Ukkonen E 1995 Online construction of suffix trees; *Algorithmica* **14** 249–260

*e*Publication: 6 August 2007