Analysis of string-searching algorithms on biological sequence databases

S. S. Sheik[1], Sumit K. Aggarwal[1], A. Poddar[2], B. Sathiyabhama, N. Balakrishnan[2] and K. Sekar[1,2,*]

[1]Bioinformatics Centre and [2]Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore 560 012, India

String-searching algorithms are used to find the occurrences of a search string in a given text. The advent of digital computers has stimulated the development of string-searching algorithms for various applications. Here, we report the performance of all string-searching algorithms on widely used biological sequence databases containing the building blocks of nucleotides (in the case of nucleic acid sequence database) and amino acids (in the case of protein sequence database). The biological sequence databases used in the present study are Protein Information Resource (PIR), SWISS-PROT, and amino acid and nucleotide sequences of all genomes available in the genome database. The average time taken for different search-string lengths considered for study has been taken as an indicator of performance for comparison between various methods.

Keywords: Pattern matching, protein sequence, nucleotide sequence, databases, string-searching algorithms.

THE problem of pattern matching or string searching is one of the oldest and most pervasive fields in computer science. Applications that require string searching can be found almost everywhere. However, recent years have witnessed immediate interest in string-searching problems, especially for rapidly growing sequence information in biology. Because of the drastic increase in incoming sequence, the demand for an efficient string-searching algorithm is well realized.

The string-searching problem is to find one or more occurrences of the interested search string within the text string. Here, we denote the search string as SStr consisting of m characters, $x_1, x_2, \ldots, x_m$ and the text string as TStr consisting of n characters, $y_1, y_2, \ldots, y_n$. All these characters are built over a finite alphabet set denoted by Σ, which has size equal to s. Literature survey revealed that several algorithms exist to perform this task efficiently and each of them adopts its own technique and methodology. Most of the algorithms that do not use the theory of automata use the concept of window to find the occurrence of SStr within TStr. A window is defined as a portion of the text whose length is equal to the length (m) of the SStr, and it slides over the text after each attempt. The current position of the window in the text is denoted by Wpos and is first initialized to zero (left ends of TStr and the window are aligned). An attempt is made to match the characters in the window with the search string characters in some predefined order. After a match, Wpos is incremented based on the skip value generated by the algorithm, in order to slide the window on the text. This procedure is repeated until the window goes to the right end of the text. However, each algorithm uses its own method to calculate the skip value.

String-searching algorithms have evolved drastically. However, the most naïve algorithm to solve this problem is to take the skip value as one. This simple approach is known as the brute force method. After this basic approach, various algorithms were developed, which improved the efficiency and each had its own advantages and limitations.

The string-searching algorithms can be addressed in different ways. In particular, they are amenable to approaches that range from the extremely theoretical to practical. On the other hand, practical implementations of the algorithm, though hard to assimilate, have proved to be more beneficial and useful. Hence, it is extremely difficult to identify a suitable and preferably fast algorithm for a given database (because of the variation in the alphabet size; for proteins the alphabet size is 20, whereas it is 4 for nucleotide sequences). In addition, only a real-time practitioner can locate and use the most appropriate algorithm to perform a particular task. Given a situation with a string-searching problem, amateur software engineers, computational biologists, researchers or students tend to dig out information through search engines (for example, Google). This approach will solve the problem, but does not guarantee that the chosen algorithm is an efficient one. Hence, we undertook this work to study the behaviour or time complexities of various algorithms on the frequently used biological sequence databases.

Many of the string-searching algorithms are implemented in two phases, viz. pre-processing phase and search phase. During the pre-processing phase, the algorithm preprocesses the search string and generates the skip value that can be used in the search phase. The pre-processed skip value helps reduce the total number of character comparisons during the search phase, thereby reducing the overall execution time. Hence, the pre-processing phase aims at simplifying the operations in the search phase. However, the efficiency of an algorithm is mainly dependent on the methodology adopted in the search phase. The search phase can be improved by altering the order in which the characters are compared at each attempt using the most appropriate skip value that maximizes the shift of the window on the text.

The need for an efficient string-searching algorithm has been realized for the databases of amino acids and nucleotides, in the post-genomics era. These databases need to be clearly mined to obtain useful information. The amount of biological sequence information generated in the recent years has seen a dramatic increase. It has been roughly estimated that the amount of biological information doubles every 20 months, and hence it is mandatory

*For correspondence. (e-mail: sekar@physics.iisc.ernet.in)

to identify an efficient and appropriate string-searching algorithm. Next, we briefly describe various existing algorithms.

There exists a myriad of algorithms for string searching. The basic approach is brute force (BF), as pointed out earlier. The first linear-time string-searching algorithm is from Morris and Pratt (MP)[1], later improved by Knuth, Morris and Pratt (KMP)[2]. Subsequent improvements over MP were done by Colussi (COLUSSI)[3], Galil Giancarlo (GG)[4] and Apostolico Crochemore (AXAMAC)[5]. The Boyer Moore (BM)[6] algorithm is considered as one of the most efficient string-searching algorithms. This has been widely recognized and used in various string-searching applications. Subsequently, several variants of the BM algorithm had appeared in the literature by introducing minor modifications. Turbo Boyer Moore (TBM)[7] algorithm is based on BM and helps increase the shift after each attempt. Tuned Boyer Moore (TUNEDBM)[8] algorithm attempts to reduce the character–character comparisons by imposing character inspections for a specified number of times (usually three). Instead of taking a shift value based on the character in the window where the mismatch occurred as in BM, Horspool (HORSPOOL)[9] algorithm always takes the last character to find the shift value. Hence, there is less number of instructions to execute at each attempt and this algorithm is more practical in approach. On the other hand, Quick Search (QS)[10] algorithm takes (always) the character placed immediately after the window to compute the shift value after each attempt. Smith (SMITH)[11] algorithm tries to maximize the shift of the window after each attempt, by taking the maximum of the shift value gained by both HORSPOOL and QS algorithm. Berry Ravindran (BR)[12] algorithm uses a two-dimensional array of the search string and works in a similar fashion like BM. The other variants of BM algorithm include Apostolico Giancarlo (AG)[13], Reverse Colussi (RC)[14] and Zhu Takaoka (ZT)[15].

There are algorithms that work on automaton with the MP or BM concept used inherently in them. The very basic algorithm uses deterministic finite automaton (DFA). Simon (SIMON)[16] algorithm also uses DFA, but also combines the traits of MP. The Forward DAWG (Directed Acyclic Word Graph) matching (FDM)[17] algorithm uses the suffix automaton of the search string. Other algorithms that use the suffix automaton are Reverse Factor (RF)[18] and Turbo Reverse Factor (TRF), which are the most efficient in practice. A variant of RF that uses the suffix oracle instead of the suffix automaton is Backward Oracle Matching (BOM)[19]. Skip Search (SKIP)[20] and KMP Skip Search (KMPSKIP)[20] algorithms use buckets to determine starting positions of the search string in the text. The other variant of RF is Backward Nondeterministic DAWG matching (BNDM)[21] algorithm, which uses bit parallelism simulation of the suffix automaton.

The Karp Rabin (KR)[22] algorithm uses hashing methodology for string searching. The algorithm Shift Or (SO)[23] is an efficient one that uses bit-wise operations for its working. Not So Naive (NSN)[24] is a string searching algorithm, which is simple in implementation. Raita (RAITA)[25] algorithm gives importance to the order of comparison at each attempt and hence is efficient in practice. The first two linear optimal space string-searching algorithms are Galil Seiferas (GS)[26] and Two Way (TW)[27]. String Matching on Ordered Alphabets is attained through SMOA[7] algorithm. Optimal Mismatch (OM)[10] and Maximal Shift (MS)[10] algorithms sort the search-string positions according to their character frequency and their leading shift.

After a careful analysis of the existing algorithms, recently Sheik–Sumit–Anindya–Balakrishnan–Sekar (SSABS)[28] proposed a new algorithm. The algorithm, SSABS, blends the advantages of QS and RAITA. In this algorithm, the order of character comparisons performed between the window and the search-string during each attempt is fixed. First, the rightmost characters of the window and the search string are compared. Secondly, the leftmost characters of the window and the search-string are compared, and then rest of the characters are compared in right to left order. In case of a mismatch in any one of the above-stated comparisons, the algorithm does not compare the remaining characters of the window. After either a match or a mismatch, the algorithm computes the shift of the window by finding the position of the bad character (character placed immediately after the window) in the search string. This shift value for all the characters in the alphabet are computed in the preprocessing phase and is used in the search phase. Hence, the algorithm SSABS is most efficient and works well in most practical situations. We now deal with about the various biological sequence databases used in the present study to identify an efficient algorithm.

Protein databases contain protein sequences and information about protein motifs and features of protein structures. The protein sequences are present in SWISS-PROT[29], PIR[30], and Protein Data Bank (PDB)[31]. SWISS-PROT provides high level of annotation, which consists of protein function, domain, protein translational modification, etc. SWISS-PROT merges all these data to minimize the redundancy of the database. It provides integration between nucleic acid sequences, protein sequences and protein tertiary structures. This database was developed and primarily maintained by Swiss Institute of Bioinformatics.

PIR is the most comprehensive and expertly annotated protein sequence database. The objective of PIR is to achieve comprehensiveness, timeliness, non-redundancy, quality annotation and complete classification. PIR data-processing involves four major steps: import, merging, classification and annotation. Once a unique protein sequence report is imported, it is assigned an accession number and is entered into PIR. Further merging, annotation and classification take place. Retrieval of the submitted or reported sequence can be performed in PIR archive. PIR is an annotated database covering the entire taxonomic range. Recently, PIR and its international partners, EBI and SIB

Table 1. Amino acids and their occurrences in various databases used in the present study

Amino acid					
Single-letter code	Three-letter code	Full name	SWISS-PROT	PIR	FAA
A	ALA	Alanine	3856544	7644886	13100890
C	CYS	Cysteine	781348	1354341	1839722
D	ASP	Aspartic acid	2630032	5065775	8295604
E	GLU	Glutamic acid	3259738	6138803	9841468
F	PHE	Phenylalanine	2011040	4034446	6335049
G	GLY	Glycine	3432262	6584906	10713539
H	HIS	Histidine	1129444	2079537	3349835
I	ILE	Isoleucine	2930055	5807964	9562897
K	LYS	Lysine	2953764	5544237	8668206
L	LEU	Leucine	4766306	9397947	15356872
M	MET	Methionine	1185715	2287852	3715491
N	ASN	Asparagine	2115899	4154656	6697619
P	PRO	Proline	2418963	4526668	6900621
Q	GLN	Glutamine	1949526	3676823	5838973
R	ARG	Arginine	2609373	5111181	8414478
S	SER	Serine	3446430	6787410	10200603
T	THR	Threonine	2730060	5301799	8319861
V	VAL	Valine	3313610	6471185	10559951
W	TRP	Tryptophan	586516	1179815	1837371
Y	TYR	Tyrosine	1546596	3020166	4820702

Table 2. Average time (10^{-2} s) taken by various algorithms for protein sequence database SWISS-PROT. Value within parenthesis represents corresponding standard deviation

Algorithm	2	4	6	8	10	12	14	16	18	20
BF	255 (2)	254 (1)	255 (1)	255 (1)	254 (1)	255 (1)	255 (1)	255 (1)	255 (0)	255 (1)
MP	75 (3)	75 (2)	77 (3)	75 (2)	75 (2)	75 (2)	76 (3)	76 (3)	76 (3)	76 (2)
KMP	74 (3)	74 (2)	75 (4)	74 (2)	75 (3)	74 (3)	74 (3)	75 (3)	74 (2)	74 (2)
COLUSSI	87 (6)	88 (1)	89 (7)	89 (7)	90 (10)	89 (7)	88 (7)	89 (10)	87 (6)	88 (9)
GG	95 (18)	95 (1)	96 (10)	98 (11)	95 (3)	94 (3)	97 (12)	98 (16)	98 (16)	94 (4)
AXAMAC	97 (3)	94 (1)	93 (2)	93 (2)	93 (2)	93 (2)	92 (1)	92 (2)	91 (1)	91 (1)
BM	72 (7)	55 (4)	51 (4)	47 (1)	46 (2)	45 (2)	44 (2)	44 (2)	43 (1)	42 (1)
TBM	122 (23)	81 (10)	67 (2)	63 (7)	58 (4)	56 (5)	54 (4)	52 (2)	50 (3)	50 (4)
AG	288 (11)	167 (3)	129 (3)	110 (4)	99 (5)	91 (4)	86 (4)	83 (5)	80 (3)	77 (3)
RC	123 (2)	82 (1)	69 (1)	62 (1)	57 (1)	55 (1)	52 (1)	51 (1)	49 (1)	48 (1)
HORSPOOL	62 (4)	50 (2)	47 (2)	45 (2)	44 (1)	43 (1)	42 (1)	42 (1)	41 (1)	41 (2)
TUNEDBM	70 (3)	56 (2)	52 (2)	49 (2)	48 (1)	47 (2)	46 (1)	46 (2)	45 (2)	44 (2)
QS	118 (3)	88 (2)	76 (2)	69 (2)	64 (2)	63 (3)	59 (2)	57 (2)	56 (2)	55 (2)
SMITH	126 (3)	93 (2)	79 (1)	71 (2)	66 (1)	62 (1)	61 (2)	58 (2)	56 (2)	55 (2)
ZT	82 (4)	59 (1)	52 (1)	48 (1)	46 (1)	44 (1)	43 (1)	43 (1)	42 (1)	42 (1)
BR	104 (1)	82 (1)	71 (1)	64 (1)	60 (1)	56 (1)	54 (1)	52 (0)	51 (0)	50 (1)
AUT	317 (18)	314 (2)	318 (23)	321 (29)	317 (18)	314 (6)	316 (18)	314 (5)	317 (19)	315 (13)
SIMON	96 (2)	96 (2)	96 (1)	96 (2)	97 (2)	96 (2)	96 (2)	96 (2)	96 (2)	95 (2)
FDM	351 (12)	395 (43)	440 (49)	473 (46)	508 (47)	518 (35)	553 (45)	569 (42)	583 (29)	605 (30)
RF	155 (11)	109 (9)	94 (8)	84 (6)	78 (4)	73 (4)	69 (4)	66 (2)	64 (3)	62 (3)
TRF	175 (15)	120 (11)	102 (7)	91 (5)	86 (7)	80 (6)	76 (4)	74 (3)	70 (4)	68 (2)
BOM	92 (15)	74 (11)	64 (2)	62 (3)	62 (3)	62 (7)	60 (5)	58 (5)	56 (2)	55 (3)
SKIP	60 (4)	55 (3)	55 (2)	55 (4)	54 (2)	52 (3)	52 (2)	52 (2)	52 (2)	52 (2)
KMPSKIP	53 (3)	50 (3)	48 (2)	47 (1)	47 (1)	46 (1)	46 (1)	46 (2)	46 (2)	45 (1)
BNDM	73 (3)	58 (2)	53 (2)	50 (2)	48 (2)	47 (1)	45 (1)	44 (1)	44 (2)	43 (1)
KR	72 (3)	69 (3)	68 (3)	67 (2)	67 (2)	67 (2)	67 (3)	67 (2)	68 (4)	66 (1)
SO	60 (5)	59 (3)	61 (6)	60 (4)	59 (3)	60 (5)	60 (5)	59 (3)	59 (3)	63 (4)
NSN	64 (6)	65 (5)	63 (7)	65 (5)	65 (5)	66 (6)	65 (5)	63 (4)	64 (5)	63 (5)
RAITA	60 (4)	50 (2)	47 (2)	44 (1)	43 (1)	43 (1)	42 (1)	42 (1)	41 (2)	41 (2)
GS	163 (4)	163 (2)	164 (2)	163 (2)	164 (3)	164 (3)	164 (2)	164 (2)	164 (3)	164 (3)
TW	73 (6)	71 (6)	70 (4)	69 (3)	69 (3)	69 (3)	69 (3)	68 (4)	69 (4)	69 (2)
SMOA	83 (7)	85 (7)	84 (7)	83 (8)	84 (7)	82 (7)	82 (7)	83 (8)	84 (7)	83 (8)
OM	60 (2)	52 (2)	49 (1)	46 (1)	45 (1)	44 (1)	44 (1)	43 (1)	43 (1)	42 (1)
MS	60 (5)	52 (4)	49 (3)	46 (1)	45 (1)	44 (1)	43 (1)	43 (2)	42 (1)	42 (1)
SSABS	**49 (3)**	**45 (2)**	**43 (1)**	**42 (1)**	**41 (1)**	**41 (2)**	**41 (2)**	**41 (2)**	**40 (1)**	**40 (1)**

Table 3. Average time (10^{-2} s) taken by various algorithms for protein sequence database PIR. Value within parenthesis represents corresponding standard deviation

Algorithm	2	4	6	8	10	12	14	16	18	20
BF	507 (1)	506 (1)	506 (1)	506 (1)	506 (1)	506 (1)	506 (1)	506 (1)	506 (1)	503 (1)
MP	157 (3)	159 (5)	158 (5)	158 (4)	159 (5)	158 (4)	158 (4)	158 (3)	158 (4)	155 (5)
KMP	157 (3)	157 (5)	158 (6)	156 (5)	157 (4)	157 (5)	157 (5)	157 (4)	156 (5)	154 (5)
COLUSSI	186 (18)	187 (14)	185 (13)	183 (1)	183 (1)	184 (14)	185 (19)	188 (26)	182 (13)	176 (2)
GG	192 (22)	196 (2)	195 (5)	197 (15)	196 (15)	197 (19)	196 (19)	198 (26)	193 (14)	192 (19)
AXAMAC	204 (6)	195 (3)	195 (5)	195 (5)	194 (5)	193 (4)	192 (3)	190 (2)	190 (2)	188 (5)
BM	150 (9)	122 (9)	112 (6)	105 (3)	102 (3)	101 (5)	100 (5)	98 (4)	98 (5)	93 (3)
TBM	240 (19)	168 (13)	144 (12)	130 (3)	124 (6)	121 (10)	118 (8)	114 (6)	112 (7)	107 (8)
AG	568 (5)	337 (6)	263 (10)	225 (5)	205 (9)	189 (5)	180 (6)	175 (8)	168 (7)	161 (6)
RC	253 (5)	172 (3)	145 (2)	132 (1)	124 (1)	118 (1)	115 (1)	112 (1)	109 (1)	104 (2)
HORSPOOL	131 (6)	110 (4)	103 (4)	99 (2)	98 (4)	95 (2)	94 (3)	94 (4)	94 (4)	89 (2)
TUNEDBM	152 (9)	121 (4)	113 (4)	107 (3)	105 (2)	104 (4)	101 (2)	101 (3)	101 (3)	96 (5)
QS	242 (4)	184 (4)	159 (3)	146 (4)	137 (3)	130 (3)	126 (2)	125 (5)	122 (5)	116 (4)
SMITH	258 (4)	192 (3)	165 (3)	150 (3)	140 (2)	133 (3)	128 (2)	125 (3)	122 (3)	116 (4)
ZT	170 (2)	127 (1)	114 (3)	106 (1)	102 (2)	99 (1)	97 (1)	95 (1)	94 (1)	91 (2)
BR	215 (3)	171 (2)	150 (1)	137 (1)	129 (1)	122 (1)	117 (1)	114 (1)	111 (1)	106 (1)
AUT	622 (3)	632 (47)	624 (14)	623 (14)	625 (33)	627 (34)	642 (63)	634 (47)	631 (46)	621 (18)
SIMON	200 (6)	200 (4)	200 (3)	199 (3)	200 (5)	200 (5)	200 (4)	199 (3)	200 (4)	195 (4)
FDM	708 (53)	781 (69)	853 (74)	921 (72)	990 (83)	1012 (70)	1089 (103)	1114 (80)	1158 (92)	1198 (77)
RF	317 (24)	222 (14)	192 (11)	175 (11)	164 (9)	153 (8)	146 (5)	141 (5)	136 (3)	128 (3)
TRF	361 (62)	243 (16)	212 (16)	191 (12)	180 (12)	167 (9)	160 (8)	154 (7)	149 (6)	140 (5)
BOM	188 (16)	155 (18)	142 (23)	135 (6)	134 (11)	131 (5)	127 (2)	126 (11)	124 (10)	117 (8)
SKIP	130 (8)	121 (8)	122 (6)	116 (4)	117 (4)	115 (6)	115 (5)	113 (4)	113 (4)	109 (4)
KMPSKIP	116 (6)	111 (7)	107 (3)	105 (2)	104 (2)	103 (3)	104 (4)	102 (3)	102 (3)	98 (2)
BNDM	156 (6)	126 (5)	115 (3)	110 (3)	105 (2)	103 (3)	101 (3)	99 (4)	98 (2)	93 (4)
KR	151 (3)	145 (3)	142 (3)	143 (4)	141 (1)	141 (2)	141 (3)	141 (3)	141 (2)	137 (3)
SO	128 (2)	127 (1)	127 (1)	127 (1)	127 (1)	126 (1)	126 (1)	127 (6)	126 (1)	125 (7)
NSN	137 (12)	138 (10)	135 (14)	140 (10)	139 (10)	140 (12)	138 (10)	134 (8)	136 (10)	132 (10)
RAITA	131 (4)	109 (3)	102 (3)	99 (3)	97 (3)	95 (1)	94 (2)	95 (4)	94 (4)	89 (4)
GS	331 (4)	330 (4)	332 (5)	330 (5)	332 (6)	331 (5)	332 (5)	333 (5)	334 (6)	329 (5)
TW	157 (12)	150 (12)	148 (8)	146 (6)	146 (7)	146 (5)	146 (6)	145 (7)	147 (8)	143 (4)
SMOA	174 (13)	177 (14)	177 (14)	173 (15)	175 (14)	171 (14)	171 (15)	173 (15)	174 (14)	170 (15)
OM	131 (3)	113 (3)	107 (2)	102 (1)	101 (2)	99 (1)	97 (1)	97 (1)	96 (1)	92 (2)
MS	129 (3)	113 (6)	106 (1)	102 (3)	100 (3)	98 (3)	96 (1)	96 (1)	95 (1)	91 (2)
SSABS	**109 (5)**	**100 (4)**	**96 (4)**	**94 (4)**	**94 (4)**	**93 (5)**	**93 (5)**	**92 (3)**	**91 (2)**	**87 (2)**

were awarded the NIH grant to produce a single worldwide database of protein sequence and function.

Information available in FAA (FASTA Amino Acid) and FNA (FASTA Nucleic Acid) corresponds to the amino acid and nucleotide sequences of various genomes (*A. thaliana*, *C. elegans*, mitochondria, *P. falciparum*, *S. cerevisiae*, bacteria, *S. pombe* and *Anopheles gambiae*) available in the genome database. These databases (FAA and FNA) have been downloaded from the National Centre of Biotechnology Information (NCBI) anonymous ftp site (ftp://ftp.ncbi.nih.gov/genbank/genomes/*).

The databases used in the present study are SWISS-PROT, PIR, FAA and FNA.

The first three databases contain amino acid sequences (alphabet size $s = 20$) and the last database has only nucleotide sequences (alphabet size $s = 4$). These databases differ substantially in size. For example, the number of protein sequences is 135,493, 283,347 and 453,861 for SWISS-PROT, PIR and FAA respectively. The number occurrences of each amino acid available in various databases is given in Table 1. It is interesting to note that none of the above databases is static and they are subject to increase as and when new sequences are available.

A total number of 837 gene sequences (comprising nucleotides; 826.31 MB size) have been deployed in the present study. This dataset contains four alphabets (nucleotides), viz. A (adenine, 239490165), C (cytosine, 183940124), G (guanine, 183818044) and T (thymine, 239419854) and hence the alphabet size is equal to four ($s = 4$). Numbers within parenthesis denote the corresponding occurrences in the entire database.

All the algorithms have been executed and tested using a 3.06 GHz processor, 1 GB of RD-RAM with 512 KB of cache memory in RedHat Linux (version 8.0). Source codes were compiled using the 'cc' compiler without any optimization. The source code for the algorithms used for comparison is taken from the literature[32]. All the programs have been executed on a single user mode to make sure that the results are more reliable.

In order to study the efficiency of the algorithms over a particular database, we have chosen databases containing amino acid and nucleotide sequences. These databases range

Table 4. Average time (10^{-2} s) taken by various algorithms for genome database (amino acids) FAA. Value within parenthesis represents corresponding standard deviation

Algorithm	2	4	6	8	10	12	14	16	18	20
BF	814 (8)	825 (83)	825 (84)	816 (11)	826 (84)	815 (12)	826 (84)	815 (8)	837 (117)	812 (1)
MP	267 (71)	265 (51)	266 (51)	265 (52)	263 (47)	256 (4)	256 (5)	263 (48)	256 (4)	254 (5)
KMP	255 (11)	254 (5)	256 (11)	254 (4)	257 (9)	254 (4)	257 (14)	254 (4)	255 (4)	253 (5)
COLUSSI	291 (19)	296 (3)	299 (25)	303 (38)	301 (29)	295 (10)	299 (30)	297 (29)	298 (36)	290 (10)
GG	306 (33)	316 (3)	314 (3)	331 (75)	329 (73)	327 (71)	311 (2)	317 (51)	317 (51)	307 (3)
AXAMAC	318 (14)	332 (97)	344 (140)	343 (126)	324 (73)	352 (134)	320 (49)	329 (113)	307 (17)	304 (20)
BM	237 (4)	189 (4)	176 (12)	167 (8)	161 (7)	159 (6)	155 (2)	153 (2)	151 (1)	152 (4)
TBM	380 (13)	302 (19)	262 (18)	236 (8)	221 (9)	208 (4)	204 (15)	198 (9)	191 (5)	178 (6)
AG	908 (8)	543 (12)	424 (12)	364 (11)	330 (11)	306 (11)	289 (9)	280 (11)	269 (9)	262 (10)
RC	403 (8)	276 (4)	233 (4)	213 (3)	199 (3)	190 (3)	184 (3)	180 (3)	175 (2)	170 (2)
HORSPOOL	211 (19)	177 (16)	166 (9)	158 (7)	155 (5)	151 (2)	150 (3)	150 (4)	148 (3)	147 (2)
TUNEDBM	236 (11)	191 (6)	178 (4)	169 (4)	166 (4)	164 (5)	160 (3)	160 (5)	159 (5)	159 (7)
QS	384 (5)	293 (5)	256 (4)	234 (4)	221 (5)	210 (4)	204 (5)	200 (5)	195 (4)	190 (5)
SMITH	412 (6)	310 (12)	269 (12)	242 (7)	227 (8)	214 (5)	206 (5)	200 (4)	195 (4)	190 (4)
ZT	279 (40)	205 (5)	182 (4)	174 (13)	166 (11)	159 (1)	158 (10)	154 (5)	152 (5)	150 (3)
BR	347 (5)	279 (4)	245 (2)	223 (2)	210 (3)	200 (2)	192 (2)	187 (3)	182 (3)	177 (3)
AUT	1002 (24)	1003 (28)	998 (21)	1003 (50)	992 (4)	993 (2)	997 (17)	994 (3)	1019 (85)	1028 (85)
SIMON	377 (126)	339 (65)	335 (46)	340 (75)	343 (73)	342 (82)	335 (58)	358 (106)	327 (6)	342 (74)
FDM	1119 (60)	1251 (100)	1373 (94)	1478 (112)	1576 (130)	1638 (119)	1724 (125)	1787 (126)	1844 (116)	1920 (137)
RF	508 (42)	363 (38)	312 (20)	282 (17)	261 (11)	248 (11)	236 (8)	227 (8)	220 (8)	211 (3)
TRF	555 (39)	390 (27)	338 (21)	306 (16)	284 (14)	267 (12)	257 (11)	248 (10)	239 (7)	232 (7)
BOM	309 (49)	247 (30)	225 (21)	216 (16)	217 (17)	208 (8)	204 (6)	198 (8)	194 (6)	189 (6)
SKIP	208 (14)	193 (11)	192 (8)	187 (6)	187 (6)	184 (6)	182 (5)	181 (4)	182 (4)	179 (5)
KMPSKIP	189 (10)	178 (7)	175 (6)	172 (4)	171 (5)	169 (4)	167 (3)	167 (3)	166 (3)	163 (4)
BNDM	250 (18)	200 (6)	187 (9)	178 (6)	171 (4)	167 (3)	164 (3)	161 (2)	158 (2)	156 (3)
KR	252 (38)	237 (29)	235 (28)	233 (28)	237 (40)	232 (27)	227 (1)	228 (1)	228 (1)	233 (34)
SO	208 (31)	209 (31)	201 (2)	206 (24)	206 (23)	207 (25)	201 (1)	207 (25)	205 (23)	202 (7)
NSN	227 (27)	226 (21)	227 (47)	234 (44)	228 (21)	243 (59)	226 (18)	224 (45)	232 (59)	220 (22)
RAITA	207 (4)	174 (3)	165 (3)	159 (3)	156 (2)	154 (3)	152 (2)	151 (3)	150 (2)	148 (2)
GS	512 (9)	510 (8)	514 (9)	518 (9)	519 (11)	521 (9)	521 (9)	525 (8)	524 (8)	524 (9)
TW	256 (19)	269 (101)	263 (68)	246 (27)	259 (66)	241 (9)	241 (10)	243 (27)	249 (36)	261 (63)
SMOA	284 (32)	285 (23)	292 (42)	283 (37)	282 (24)	278 (32)	276 (26)	283 (35)	281 (24)	281 (33)
OM	206 (6)	179 (4)	171 (14)	164 (13)	160 (11)	156 (3)	154 (2)	154 (7)	151 (2)	152 (5)
MS	208 (5)	183 (8)	175 (6)	167 (3)	164 (5)	160 (4)	157 (2)	157 (3)	156 (3)	152 (2)
SSABS	**171 (8)**	**159 (5)**	**155 (5)**	**151 (4)**	**149 (4)**	**147 (3)**	**147 (3)**	**146 (2)**	**145 (2)**	**144 (3)**

from alphabet sizes that are small (in the case of nucleotides) to those that are large (in the case of amino acids). SWISS-PROT, PIR and genome database of FAA contain amino acid sequences (hence $\sum$ = (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y) and $s = 20$) and FNA genome database contains nucleotide sequences (hence $\sum$ = (A, C, G, T) and $s = 4$). No prior pre-processing has been done on the databases used in the present analysis. We have tested each of the algorithms against several search-string lengths (2, 4, 6, 8, 10, 12, 14, 16, 18 and 20). In addition, for each search-string length, 50 randomly selected search strings were used. The above-mentioned procedure is repeated for all the databases deployed in the present study. Tables 2–5 show the average time taken by various algorithms on the databases SWISS-PROT, PIR, FAA genome database and FNA genome database. The reported average time is 10^{-2} s. The following points have been derived after careful examination of the time taken by various algorithms reported in Tables 2–5.

First, in the case of SWISS-PROT, the algorithm SSABS, performs better for search strings of varying lengths. HORSPOOL and RAITA show better timings as also ZT, KMPSKIP and MS in a few instances. Secondly, in the case of the PIR database, SSABS shows better timings over other algorithms followed by HORSPOOL and RAITA. Thirdly, in the case of huge protein sequence database derived from all available genomes, SSABS works consistently better for varying lengths of search string, followed by HORSPOOL and RAITA. At the outset, SSABS, HORSPOOL and RAITA work well in the case of the protein sequence databases ($s = 20$). However, in the case of a nucleotide sequence database ($s = 4$), SSABS works well for search-string length up to six. It is interesting to note that for longer search strings ($m \geq 6$), the ZT performs better, followed by BNDM.

To conclude, the algorithm SSABS performs well on protein sequences ($s = 20$), irrespective of the size of the database and the length of the search string. The algorithm ZT

Table 5. Average time (10^{-2} s) taken by various algorithms for genome database (nucleotides) FNA. Value within parenthesis represents corresponding standard deviation

Algorithm	2	4	6	8	10	12	14	16	18	20
BF	4554 (53)	4484 (23)	4494 (49)	4494 (36)	4488 (29)	4497 (39)	4492 (39)	4494 (31)	4487 (34)	4487 (38)
MP	1620 (49)	1627 (150)	1608 (98)	1616 (100)	1621 (150)	1602 (96)	1631 (97)	1598 (50)	1588 (49)	1603 (86)
KMP	1751 (508)	1677 (325)	1585 (185)	1644 (212)	1652 (313)	1614 (181)	1695 (343)	1586 (69)	1573 (133)	1602 (119)
COLUSSI	1712 (154)	1650 (70)	1631 (153)	1681 (89)	1659 (142)	1677 (136)	1652 (144)	1632 (91)	1638 (136)	1655 (104)
GG	1704 (302)	1746 (74)	1709 (141)	1775 (87)	1728 (109)	1769 (132)	1724 (114)	1735 (109)	1727 (138)	1746 (90)
AXAMAC	1972 (73)	1808 (266)	1729 (221)	1752 (76)	1726 (136)	1825 (324)	1733 (119)	1723 (111)	1737 (240)	1760 (213)
BM	1643 (282)	1305 (215)	1180 (170)	1104 (109)	1093 (106)	1080 (86)	1029 (70)	1060 (70)	1042 (63)	1010 (79)
TBM	2530 (416)	2155 (314)	1817 (157)	1800 (469)	1737 (452)	1697 (286)	1573 (207)	1694 (315)	1602 (211)	1438 (183)
AG	5771 (354)	3919 (358)	3139 (341)	2890 (361)	2825 (398)	2837 (484)	2601 (436)	2789 (388)	2684 (391)	2512 (313)
RC	2210 (51)	1608 (46)	1372 (45)	1249 (45)	1182 (35)	1129 (28)	1096 (33)	1053 (34)	1033 (38)	1007 (32)
HORSPOOL	1809 (136)	1372 (172)	1226 (102)	1194 (123)	1213 (196)	1229 (187)	1194 (174)	1224 (149)	1213 (177)	1174 (110)
TUNEDBM	1967 (280)	1458 (103)	1328 (137)	1283 (125)	1275 (155)	1305 (165)	1245 (158)	1291 (135)	1281 (164)	1252 (129)
QS	2477 (100)	2064 (177)	1893 (223)	1803 (204)	1838 (248)	1845 (259)	1716 (251)	1858 (279)	1859 (225)	1724 (235)
SMITH	2720 (107)	2113 (147)	1872 (203)	1756 (194)	1766 (272)	1798 (292)	1699 (255)	1804 (278)	1760 (236)	1565 (451)
ZT	1728 (42)	1233 (40)	1061 (26)	**985 (26)**	**951 (47)**	**916 (25)**	**896 (25)**	**873 (26)**	**868 (24)**	**855 (27)**
BR	2272 (81)	1764 (105)	1568 (102)	1429 (88)	1352 (87)	1297 (61)	1236 (66)	1206 (76)	1203 (62)	1169 (69)
AUT	5553 (53)	5436 (22)	5430 (52)	5444 (85)	5439 (71)	5437 (57)	5447 (89)	5429 (27)	5423 (26)	5418 (17)
SIMON	2195 (502)	2094 (316)	2081 (316)	2076 (266)	2079 (290)	2030 (63)	2090 (274)	2107 (314)	2075 (233)	2077 (184)
FDM	8620 (861)	10449 (923)	11432 (996)	12191 (972)	12955 (638)	13233 (722)	13465 (466)	13595 (702)	13711 (151)	13865 (115)
RF	5418 (532)	3624 (244)	2886 (193)	2488 (165)	2255 (93)	2064 (94)	1913 (73)	1811 (70)	1714 (47)	1640 (42)
TRF	5905 (803)	4347 (380)	3499 (295)	3013 (231)	2718 (120)	2461 (131)	2278 (107)	2147 (99)	2024 (70)	1953 (59)
BOM	2363 (399)	1715 (123)	1459 (63)	1326 (65)	1239 (43)	1197 (54)	1136 (47)	1124 (94)	1081 (66)	1054 (52)
SKIP	2168 (221)	1866 (93)	1814 (66)	1771 (84)	1765 (71)	1735 (61)	1740 (66)	1728 (73)	1736 (74)	1719 (50)
KMPSKIP	1677 (192)	1518 (82)	1430 (51)	1408 (84)	1380 (93)	1357 (68)	1348 (40)	1346 (56)	1325 (40)	1325 (70)
BNDM	1890 (268)	1401 (94)	1201 (73)	1076 (51)	1008 (36)	970 (35)	932 (42)	908 (38)	880 (31)	862 (34)
KR	1532 (109)	1233 (29)	1210 (20)	1195 (23)	1188 (23)	1194 (23)	1199 (24)	1195 (25)	1190 (25)	1195 (22)
SO	1182 (121)	1078 (102)	1075 (104)	1065 (28)	1064 (25)	1092 (133)	1071 (107)	1075 (102)	1094 (130)	1078 (111)
NSN	1800 (183)	1782 (138)	1741 (189)	1768 (148)	1743 (148)	1788 (155)	1750 (133)	1770 (180)	1743 (189)	1788 (138)
RAITA	1612 (177)	1200 (106)	1097 (102)	1058 (96)	1033 (77)	1051 (101)	1043 (109)	1056 (99)	1046 (118)	1021 (81)
GS	3448 (101)	3369 (102)	3370 (95)	3384 (98)	3359 (94)	3374 (95)	3413 (85)	3396 (107)	3364 (94)	3381 (100)
TW	1664 (204)	1479 (154)	1414 (174)	1364 (106)	1382 (149)	1387 (87)	1404 (134)	1416 (102)	1396 (129)	1403 (154)
SMOA	2383 (320)	2497 (301)	2402 (294)	2488 (368)	2445 (298)	2419 (263)	2498 (473)	2540 (426)	2497 (331)	2545 (403)
OM	1464 (101)	1256 (79)	1143 (71)	1088 (76)	1074 (68)	1068 (68)	1016 (59)	1026 (62)	1022 (67)	999 (51)
MS	1457 (172)	1240 (137)	1104 (76)	1061 (92)	1020 (42)	1008 (64)	957 (41)	985 (84)	957 (47)	940 (66)
SSABS	**1135 (43)**	**1063 (60)**	**1020 (93)**	994 (95)	1005 (96)	1026 (137)	984 (98)	996 (72)	1026 (140)	987 (103)

performs well in the case of nucleotide sequences ($s = 4$), when the search-string length is more than six. However, SSABS provides reasonable competition over other algorithms for search-string lengths less than or equal to six. To gain further understanding, our future interest is to analyse the performance of various algorithms in the English language, where the alphabet size is more than 26.

1. Morris, Jr. J. H. and Pratt, V. R., Tech. Rep. 40, University of California, Berkeley, 1970.
2. Knuth, D. E. *et al*., *SIAM J. Comput.*, 1977, **6**, 323–350.
3. Colussi, L., *Inf. Comput.*, 1991, **95**, 225–251.
4. Galil, Z. and Giancarlo, R., *SIAM J. Comput.*, 1992, **21**, 407–437.
5. Apostiloco, A. and Crochemore, M., *Inf. Comput.*, 1991, **95**, 76–95.
6. Boyer, R. S. and Moore, J. S., *Commun. ACM*, 1977, **20**, 762–772.
7. Crochemore, M., *Theor. Comput. Sci.*, 1992, **92**, 33–47.
8. Hume, A. and Sunday, D. M., *Software – Practice Experience*, 1991, **21**, 1221–1248.
9. Horspool, R. N., *Software – Practice Experience*, 1980, **10**, 501–506.
10. Sunday, D. M., *Commun. ACM*, 1990, **33**, 132–142.
11. Smith, P. D., *Software – Practice Experience*, 1991, **21**, 1065–1074.
12. Berry, T. and Ravindran, S., In Proceedings of the Prague Stringology Club Workshop ’99 (eds Holub, J. and Simánek, M.), Collaborative Report DC-99-05, Czech Technical University, Prague, Czech Republic, 1999, pp. 16–26.
13. Apostolico, A. and Giancarlo, R., *SIAM J. Comput.*, 1986, **15**, 98–105.
14. Colussi, L., *J. Algorithms*, 1994, **16**, 163–189.
15. Zhu, R. F. and Takaoka, T., *J. Inf. Process.*, 1987, **10**, 173–177.
16. Simon, I., In Proceedings of 1st American Workshop on String Processing (eds Baeza-Yates, R. A. and Ziviani, N.), Universidade Federal de Minas Gerais, Brazil, 1993, pp. 151–157.
17. Crochmore, M. and Rytter, W., Proceedings of the 7th Annual ACM Conference on Computational Learning Theory, 1994, pp. 3–11.
18. Lecroq, T., *Theor. Comput. Sci.*, **92**, 119–144.
19. Allauzen, C. *et al*., In Proceedings of SOFSEM’99, Theory and Practice of Informatics (eds Pavelka, J., Tel, G. and Bartosek, M.), *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1999, vol. 1725, pp. 291–306.
20. Charras, C. *et al*., *Lecture Notes in Computer Science* (ed. Farach-Colton, M.), Springer-Verlag, Berlin, 1998, vol. 1448, pp. 55–64.

21. Navarro, G. and Raffinot, M., *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1998, vol. 1448, pp. 14–31.
22. Karp, R. M. and Rabin, M. O., *IBM J. Res. Dev.*, 1987, **31**, 249–260.
23. Baeza-Yates, R. A. and Gonnet, G. H., *Commun. ACM*, 1992, **35**, 74–82.
24. Hancart, C., *Actes des 2^e Journées Franco-Belges* (ed. Krob, D.), 1991, PUR 176, Rouen, France, 1992, pp. 99–110.
25. Raita, T., *Software – Practice Experience*, 1992, **22**, 879–884.
26. Galil, Z. and Seiferas, J., *J. Comput. Syst. Sci*., 1983, **26**, 280–294.
27. Crochemore, M. and Perrin, D., *J. ACM*, 1991, **38**, 651–675.
28. Sheik, S. S., Aggarwal, S. K., Poddar, A., Balakrishnan, N. and Sekar, K., *J. Chem. Inf. Comp. Sci.*, 2004, **44**, 1251–1256.
29. Bairoch, A. and Apweiler, R., *Nucleic Acids Res.*, 1998, **26**, 38–42.
30. Barker, W. C. *et al*., *Nucleic Acids Res.*, 1998, **28**, 27–32.
31. Bernstein, F. C. *et al.*, *J. Mol. Biol*., 1977, **112**, 535–542.
32. Charras, C. and Lecroq, T., *Handbook of Exact String Matching Algorithms* (available in the website: http://www-igm.univ-mlv.fr/~lecroq/string/).

ACKNOWLEDGEMENTS. We thank the Bioinformatics Centre (DIC), the Interactive Graphics Based Molecular Modelling facility (IGBMM) and the Supercomputer Education and Research Centre, IISc, Bangalore for support. This work is completely supported by the Institute-vide Computational Genomics Project supported by the Department of Biotechnology (DBT), New Delhi. DIC and IGBMM are supported by DBT.

Received 20 July 2004; revised accepted 1 February 2005

Compositional symmetry of DNA duplex in bacterial genomes

Sujit Kumar Verma[1,*], Debojyoti Das[1], Siddharatha Sankar Satapathy[2], Alak Kumar Buragohain[1] and Suvendra Kumar Ray[1]

[1]Department of Molecular Biology and Biotechnology, and
[2]Department of Computer Science and Information Technology, Tezpur University, Tezpur 784 028, India

Computational analysis of seven bacterial genomes revealed that both the DNA strands in a genome exhibit compositional symmetry in terms of the abundance of a non-palindromic oligonucleotide (di, tri, tetra, penta, hexa, hepta and octa). This symmetry in DNA duplex suggests that both strands in the duplex possess similar compositional characteristics, though the nucleotide sequences in the DNA strands are different (complementary). This compositional symmetry between DNA strands in genomes may be due to the abundance of coding sequences in both the strands and to ensure the synchronous completion of replication of the two strands.

Keywords: Genome, DNA duplex, non-palindromic sequence, oligonucleotide composition, coding sequences, DNA replication.

A chromosome is made up of a DNA duplex in which the two DNA strands are antiparallel and complementary to each other: adenine (A) of one strand pairs with thymine (T) of the other strand, and guanine (G) of one strand pairs with cytosine (C) of the other strand[1]. Thus sequences of both the strands are different, except at the palindromic regions (symmetrical DNA sequence). In the case of a palindromic oligonucleotide, its abundance in both the DNA strands in a genome is identical. However, in the case of a non-palindromic oligonucleotide (non-symmetrical DNA sequence), its abundance in both the DNA strands in a genome might be different. In this study we have analysed seven bacterial genomes (*Bacillus subtilis*[2], *Escherichia coli*[3], *Haemophilus influenzae*[4], *Pseudomonas aeruginosa*[5], *Pseudomonas syringae*[6], *Ralstonia solanacearum*[7] (mega plasmid and chromosome) and *Xanthomonas campestris* pv. *campestris*[8] (*Xcc*)) for abundance of non-palindromic oligonucleotides in both the DNA strands. We present evidence that abundance of a non-palindromic oligonucleotide in both the DNA strands in a genome is similar. This compositional symmetry of the DNA duplex in genomes is interesting and suggests that there is a tendency in the genome to maintain similarity between both the DNA strands, though functionally the two strands have different attributes.

Complete genome sequences of *B. subtilis*, *E. coli*, *H. influenzae*, *P. aeruginosa*, *P. syringae*, *R. solanacearum* and *Xcc* were downloaded from the 'genome information broker site' (www.gib.genes.nig.ac.jp). These sequences were analysed for the occurrence of non-palindromic oligonucleotide sequences (di, tri, tetra, penta, hexa, hepta and octa nucleotides; Table 1) in both DNA strands using a computer program 'seqsearch' (developed by the authors). Since DNA strands are complementary to each other, by studying nucleotide composition of one of the strands, composition of the other strand can be determined, e.g. the number of As present in one of the strands is equal to the number of Ts present in the other strand, and the number of Gs present in one strand is equal to the number of Cs present in the other strand. Similar logic can also be applied for studying oligonucleotide composition as well, e.g. total number of TG dinucleotides present in one of the strands of a genome is same as the total number of CA dinucleotides present in the other strand of the genome. Thus if we count the total number of TG and CA in one of the strands of the genome, then we would be able to compare between the number of TGs present in both the DNA strands. In this study, we have analysed one of the DNA strands in the genomes for comparing the abundance of nucleotides/oligonucleotides

*For correspondence. (e-mail: suven@tezu.ernet.in)