A Parallel Pairwise Local Sequence Alignment Algorithm

Sanghamitra Bandyopadhyay, *Senior Member, IEEE*, and Ramkrishna Mitra*

***Abstract*—Researchers are compelled to use heuristic-based pairwise sequence alignment tools instead of Smith–Waterman (SW) algorithm due to space and time constraints, thereby losing significant amount of sensitivity. Parallelization is a possible solution, though, till date, the parallelization is restricted to database searching through database fragmentation. In this paper, the power of a cluster computer is utilized for developing a parallel algorithm, RPAlign, involving, first, the detection of regions that are potentially alignable, followed by their actual alignment. RPAlign is found to reduce the timing requirement by a factor of upto 9 and 99 when used with the basic local alignment search tool (BLAST) and SW, respectively, while keeping the sensitivity similar to the corresponding method. For distantly related sequences, which remain undetected by BLAST, RPAlign with SW can be used. Again, for megabase-scale sequences, when SW becomes computationally intractable, the proposed method can still align them reasonably fast with high sensitivity.**

***Index Terms*—Basic local alignment search tool (BLAST), message passing interface (MPI), parallel computing, Smith–Waterman (SW).**

I. Introduction

PAIRWISE sequence alignment is a challenging task because of the exponential growth of genomic information, necessitating large-scale comparison of two input strings. The size of GenBank/the European Molecular Biology Laboratory (EMBL)/the DNA DataBank of Japan (DDBJ) nucleotide database is now doubling in every 15 months [1]. To search databases to find out sequences similar to a given query sequence, the search programs compute an alignment score for every sequence in the database. This score represents the degree of similarity between the query and database sequence. A dynamic programming algorithm for computing the optimal local alignment score was first described by Smith and Waterman [2], and later improved in [3] for linear gap penalty functions. Though dynamic programming is the best alignment procedure so far, but it is not suitable for large strings in terms of both time and space. For two strings of lengths m and n, the time and space complexities of the Smith and Waterman (SW) algorithm are $O(mn)$. The time and space complexity had been improved to $O(rn)$ in [4], where r is the amount of allowed error, by considering only the useful part of the distance matrix. However, for large error rates, r is $O(m)$, so the complexity is still $O(mn)$. Later on, the space complexity of SW was improved to $O(n)$ [5]. Dynamic programming has been accelerated through GLASS by first finding exactly matching long substrings, but the time and space complexity are still high [6]. LAGAN [7] is another implementation of dynamic programming, but is not applicable on a genome scale without prior information ("anchors") that directs comparison to orthologous regions.

There are many heuristic-based search tools, and they can be categorized into hash-table-based search tools and suffix-tree-based tools. FASTA [8], basic local alignment search tool (BLAST) [9], [10], MegaBLAST [11], BL2SEQ [12], WU-BLAST [13], PipMaker [14], Pattern Hunter [15], BLAT [16], and SSAHA [17] are methods that belong to the category of hash-table-based tools. These are basically achieved by "seed-and-extend" methods. Current hash-table-based search tools handle short queries well, but become very inefficient, in terms of both time and space, for long queries. The limitation of seed-and-extend methods has been overcome in [18]–[20]. In [20], a parallel technique called Pash was designed to compare genome-sized datasets. However, it is not the best choice when indels are prevalent. As mentioned in [20], Pash is relatively inefficient when mapping a relatively small dataset onto a relatively larger one.

Suffix tree is another efficient approach on which various search tools have been developed. These include MUMmer [21], QUASAR [22], REPuter [23], and AVID [24]. There are many significant problems with the suffix-tree-based approach: they manage mismatches inefficiently (they are good for highly similar strings, but fail to recognize more distant homologies) and they have a high space overhead.

Recent advances in parallelization make it possible to implement BLAST (http://www.ncbi.nlm.nih.gov/BLAST) in a parallel setup as well. Earlier works on parallel sequence search mostly focus on distributing the query set across several cluster nodes [25]–[27], each of which executes a serial job. Throughput is increased, but the time for a particular query to complete is unchanged. Other existing parallel techniques have been focusing mostly on database segmentation. In this approach, the database is partitioned among cluster nodes and an assigned part of the database is searched for the same query [28], [29]. This approach of database splitting was developed in the mpiBLAST [30]. Among the several published parallel BLAST codes, mpiBLAST reported the highest speedup, underwent the largest scalability tests, and has been directly integrated with the National Center for Biotechnology Information (NCBI) toolkit.

Manuscript received June 25, 2008. First published April 10, 2009; current version published July 31, 2009. The work was supported by the Department of Science and Technology, Government of India, under the Swarnajayanti Fellowship Scheme, under Grant DST/SJF/ET-02/2006-07. *Asterisk indicates corresponding author.*

S. Bandyopadhyay is with the Machine Intelligence Unit, Indian Statistical Institute, Kolkata 700108, India (e-mail: sanghami@isical.ac.in).

*R. Mitra is with the Machine Intelligence Unit, Indian Statistical Institute, Kolkata 700108, India (e-mail: rmitra_t@isical.ac.in).

Digital Object Identifier 10.1109/TNB.2009.2019642

A subsequent efficient algorithm pioBLAST [31] was developed that has reduced nonsearch overheads of mpiBLAST by focusing on the use of collective I/O and dynamic database partitioning. The other useful works on the parallelization of BLAST are ParAlign [32], pp-Blast [33], and ScalaBLAST [34]. All the aforementioned parallel BLAST implementations are based on searching database sequences in parallel by segmenting and distributing the set of query sequences or database sequences.

Although BLAST is widely used in the bioinformatics community, it is well known to suffer from low sensitivity as compared to SW. In particular for the sequences, which are distantly related, BLAST may be unable to throw up any hit, a problem that SW can overcome. However, SW is known to be unable to compare two large DNA sequences due to its computational complexity. Some attempts in developing faster, parallel implementations of the SW algorithm can be found in [35] and [36], but these are essentially database searching algorithm. In [36], a vector implementation of SW makes the rigorous SW competitive with BLAST (within a factor of 5 or less), but for large-scale DNA sequence, it is not practical. Few attempts have been made for developing parallel algorithms for comparing a pair of large-scale sequences. This requires proper fragmentation of the two sequences, and distribution of the fragments to the different nodes of a parallel computer. Not much work is available in this direction probably because it has been difficult to parallelly identify those subsequences that are actually alignable in the two sequences, though some sequential algorithms have been attempted in this regard [19].

In this paper, we propose an efficient algorithm, which can overcome this problem and can align two DNA or protein sequences in parallel by identifying regions that are potentially alignable (RPAs). Once this is done parallelly, the task of aligning these subsequences can be easily parallelized, resulting in a gain in computation time. Such a parallel algorithm, referred to as RPAlign, is developed in this paper.

It employs frequency counts in windows to detect the RPAs in the two subsequences. A cluster computer is utilized for implementing RPAlign using message passing interface (MPI). Note that our task is not to propose a new alignment algorithm, but to improve the time requirement, through the use of judicious parallelism, of any pairwise local sequence alignment method.

II. Systems and Methods

The code is written in C using MPI. A cluster of 18 nodes is used with Linux WS 3.0 standard operating system. Master node consists of Intel Xeon 2.8 GHz single CPU and 1 GB RAM. Each slave or worker node consists of Pentium IV 2.8 GHz CPU and 512 MB RAM. The bl2seq module of NCBI BLAST toolkit (version 2.2.15) [10] and SW implementation, ssearch34 (downloaded from the Web site of the University of Virginia), are used for both DNA and protein sequence comparison.

III. Algorithm and Implementation

The detection of RPA between two DNA or protein sequences is based on the computation of the frequency of each type of element. The system incorporates one master processor (MP) and $n-1$ worker processors (WPs). The proposed algorithm is described shortly in detail.

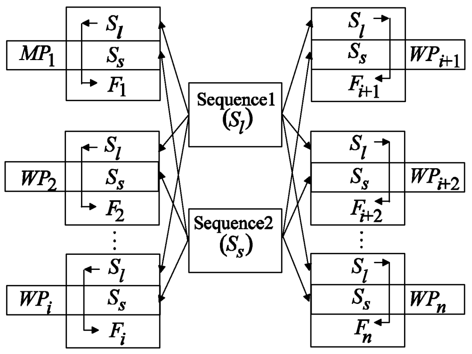

Fig. 1. Parallel I/O and dynamic partition of the larger sequence (here, S_l).

A. *Efficient Data Handling for Parallel Processing*

The MP and WPs parallelly read the two input sequences S_l and S_s, assuming $|S_l| > |S_s|$, and determine their lengths. Each processor (including the MP, which is treated as WP_1 in the following discussion) then extracts one overlapping subsequence from the larger sequence S_l. Considering that the length of the overlapping window is denoted by w, the length of each subsequence or fragment F_i, $i = 1, 2, \ldots, n$, is given by

$$\frac{|S_l|}{n} + w$$

where n is the number of nodes in the cluster (see Fig. 1). Therefore, the start and end positions of fragment F_i, denoted by Start_i and End_i, respectively, are given by

$$\mathrm{Start}_i = (i-1) \times \frac{|S_l|}{n} + 1$$

and

$$\mathrm{End}_i = i \times \frac{|S_l|}{n} + w.$$

Parallel file I/O in a shared memory framework is used through which load balancing is performed and copying overhead is reduced.

B. *Computing Frequencies and Composite Scores*

The tasks performed by processor P_i, $i = 1, 2, \ldots, n$, are outlined in Fig. 2. These are now described in detail.

F_i, the fragment of S_l read by P_i, is further divided into substrings of length w by sliding it one letter at a time to generate substrings F_{ij}, $j = 1, 2, \ldots, |Fi| - w + 1$. The second sequence S_s is also divided into substrings of length w by shifting w letters at a time to yield S_{sk} substrings, where $k = 1, 2, \ldots, \lceil |S_s|/w \rceil$. Then, for every possible substring F_{ij} or S_{sk}, the frequencies of each type of element are determined as $f_e(F_{ij})$ or $f_e(S_{sk})$ for $e = 1, 2, \ldots, 5$ for DNA and $e = 1, 2, \ldots, 20$ for protein sequence. For the DNA sequence, elements are A, T, G, C, and N, where N stands for the unknown, and for protein sequence, there are 20 different types of amino acids present. A score is then computed, which is called composite score (CS), on the basis of f for a substring pair. On the

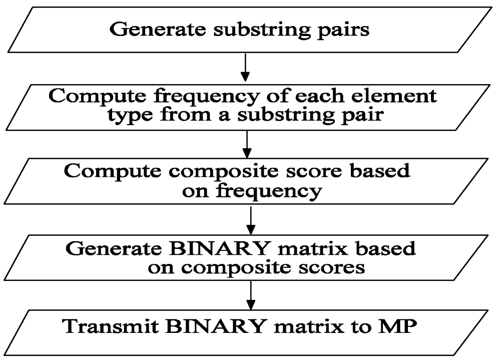
Generate substring pairs
Compute frequency of each element type from a substring pair
Compute composite score based on frequency
Generate BINARY matrix based on composite scores
Transmit BINARY matrix to MP

Fig. 2. Flowchart of BINARY matrix formation in each processor.

basis of this score, RPA will be detected. As protein sequences are more complex in nature than the DNA sequences, and as substitution matrix plays an important role for the alignment, computation of RPA for protein sequences is much more complicated than for DNA sequences. This is first described in the following. On the other hand, as DNA sequences are usually much larger than protein sequences, efficient data handling is essential. For this reason, an optimization technique has been developed.

1) Protein Sequences: For protein sequences, two different substitution matrices, BLOSUM62 and BLOSUM50 [37], are used, though other standard substitution matrices can also be implemented. After computing the frequency *fe*, $e = 1, 2, \ldots, 20$, of all the substrings, CSs are now computed as follows:

$$\mathrm{CS}(F_{ij}, S_{sk}) = \sum_{e=1}^{20} \mathrm{Min}(f_e(F_{ij}), f_\mathrm{e}(S_{sk})) \times \mathrm{Mat\_Score}(A_e, A_e) \tag{1}$$

where Mat_Score is the score for the two identical elements (A_e) given by the substitution matrix.

2) DNA Sequences: Here, the minimum frequencies of A, T, G, C, and N, corresponding to each pair F_{ij} and S_{sk}, are computed. On the basis of the frequencies f_e, $e = 1, 2, \ldots, 5$, CS is now computed as

$$\mathrm{CS}(F_{ij}, S_{sk}) = \sum_{e=1}^{5} \mathrm{Min}(f_e(F_{ij}), f_\mathrm{e}(S_{sk})). \tag{2}$$

C. Generating the BINARY Matrix

It may be noted that CS is a gross overestimation of the actual alignment score of the two substrings. This is done to ensure that even after such an overestimation, if CS $< \theta$, where θ is a threshold value, then the corresponding substrings need not be considered as they are not alignable. Now, a matrix called BINARY of dimension $\lceil |S_s|/w \rceil \times \lceil |F_i|/w \rceil$ is generated in node i, where each row and column represents w length of nonoverlapping letters of S_s and F_i, respectively. Initialize BINARY matrix to all 0's. For each S_{sk}, $k = 1, 2, \ldots, \lceil |S_s|/w \rceil$, w consecutive fragments from F_i are used to compute w different CS values. If any of these CS values exceeds a threshold θ, then cells (k, j) and $(k, j+1)$ of BINARY matrix are set to 1

```
For $k$= 1, 2, …, $\lceil |S_s|/w \rceil$
  For $j$=1, 2, …, $\lceil |F_i|/w \rceil$
    Compare $F_{ij}$, $j$= ($j$-1)$w$+1, …, $jw$,
    and $S_{sk}$ to provide $CS^1$, $CS^2$, …, $CS^w$.
    If max ($CS^k$, $k$=1,…, $w$) > $\theta$ then
      set $BINARY(k, j) = BINARY(k, j+1) = 1$
    End If
  End
End
```

Fig. 3. Computation of BINARY matrix in processor P_i.

as w consecutive CS values cover $2w - 1$ letters. Fig. 3 states this process formally.

Note that all the w CS values need not be computed. As soon as a value exceeding θ is obtained, the remaining substring pairs are not considered any further. Since DNA sequences can be extremely long, leading to high computational cost for comparing all the CS values with θ, a procedure for optimizing this computation is described shortly.

The substring F_{ij+1} is generated by sliding F_{ij} by one letter to the right, i.e., to generate a new substring F_{ij+1}, one letter is removed from the leftmost position and one new letter is inserted at the rightmost position of the current substring. It can generate two possible effects on F_{ij+1}.

1) If the inserted and deleted letters are the same, then f vector of F_{ij} and F_{ij+1} are the same.
2) If the inserted and deleted letters are not identical, then the frequency of one element type (A, T, C, G, or N) is decreased by one, and the other one is increased by one.

On the basis of the previous logic, it is clear that CS values for DNA sequences can be changed by at most 1. Now, for computing the value in cell (k, j) and $(k, j+1)$ of the BINARY matrix, CS is first computed between S_{sk} and F_{ij}. If CS $>$ θ, then as before, computation is discontinued and BINARY (k, j) = BINARY$(k, j+1) = 1$. However, if CS $< \theta$, then let $\theta' = \theta -$CS. In this case, the next comparison needs to be made between $F_{i(j+\theta'+1)}$ and S_{sk}. In this way, when $(j + \theta' + 1) >$ w, then the search is discontinued, since under no circumstance can any of the CS value exceed θ. If $(j + \theta' + 1) < w$, then the process is repeated.

D. Merging and Redistribution

After each WP$_i$ completes the computation of the BINARY matrix, these are transmitted to the MP. Here, the matrices are collated side by side to yield a matrix called RPA_DETECTION matrix. In this matrix, diagonals that are strings of all 1's are found. The start and end positions of these diagonals define the RPAs. Note that there is a possibility for multiple overlapping surfaces of similarity (diagonals) in RPA_DETECTION matrix. As a result, there is a chance that a particular sequence fragment may be included in multiple aligned segments. In RPAlign, this problem is reduced by merging the adjacent overlapping diagonals. But the overlapping diagonals, which are not adjacent, are also considered as RPAs, the reason being that prior knowledge about which RPA will provide the best score is absent. The diagonal, which is a subset of another diagonal, is not considered as

TABLE I
NUCLEOTIDE SEQUENCE PAIRS AND THEIR RESPECTIVE IDS AND LENGTHS

Pair	Sequence 1		Sequence 2	
	GI Number	Length(bp)	GI Number	Length (bp)
P1	92296557	723	114157166	1549
P2	118562368	6485	89142743	6736
P3	13273284	16571	508206	13246
P4	28876381	38206	303969	34214
P5	209811	35937	28876316	41796
P6	28876316	41796	28876437	40014
P7	112806880	44237	114804244	158484
P8	41179002	203828	114804244	158484

TABLE II
PROTEIN SEQUENCE PAIRS AND THEIR RESPECTIVE IDS AND LENGTHS

Pair	Sequence 1		Sequence 2	
	GI Number	Length(bp)	GI Number	Length (bp)
R1	28195689	3212	125846343	8697
R2	71480173	8026	119568112	8757
R3	28195689	3212	119568124	8797
R4	Set10 eu	6015	125983774	8130
R5	d1allb a.1.1.3	161	d1wmub a.1.1.2	146
R6	d1jbob a.1.1.3	172	d3sdha a.1.1.2	145
R7	119568124	8797	125983774	4181
R8	119568112	8757	118085751	3728
R9	119568122	8779	125853858	3461
R10	47221249	6015	125983774	4181

an RPA. Finally, all these RPAs are redistributed to each node, including the MP, in such a way that the load balancing will be achieved. Thereafter, each node performs the actual alignment either by BLAST (providing RPAlign BLAST) or by SW implementation (providing RPAlign SW), and alignment output is stored in a shared memory space.

IV. RESULTS

The speed and, more importantly, the quality of alignment of the proposed method is evaluated using a set of ten protein and eight DNA sequence pairs, and seven megabase-scale DNA sequence pairs. The lengths of the input sequences range from 145 to 8797 residues for protein sequences and from 723 bp to 11.1 mb for DNA and megabase-scale DNA sequences. The DNA sequences are described in Table I and the protein sequences are described in Table II. The window length w is chosen for DNA sequences in general, as min $(|S_l|/40, 25\,000)$ and for protein sequences of length >3 kb, $w = 500$ is recommended. The variation of the performance of RPAlign is studied for different values of the threshold θ in terms of the window size (or, different values of θ/w). To ensure that the proposed method provides high quality of alignment in terms of speed and sensitivity, a number of experiments have been carried out using different alignment parameters.

The first two experiments (Tables III and IV) show the sensitivity comparison of DNA sequence pairs (P1–P8) using SW, RPAlign SW, BLAST, and RPAlign BLAST. The experiments are based on DNA matrix $+5/-4$, and two different gap penalties (gap open (G/O) and gap extension (G/E) penalties) (details of the input parameters are given in each table). Table V shows the timing requirement of SW, RPAlign SW, BLAST, and RPAlign BLAST based on the parameters mentioned in Table III. It also shows the time gain (TG) attained by RPAlign SW and RPAlign BLAST with respect to SW and BLAST, respectively. Timing comparison based on the parameters of the second experiment (shown in Table IV) is not given as the required time is approximately the same.

To measure the sensitivity for protein sequences, ten pairs are considered (R1–R10), out of which R5–R10 are distantly related protein sequences and have significantly less homology. Specially, R5 and R6 are taken from ASTRAL database (http://astral.berkeley.edu/), where sequences have less than 40% similarity in nature [38] (see Table VII). For R1–R4, the experiments are done based on the substitution matrices BLOSUM62 and BLOSUM50 (see Table VI). For R5–R10, the experiments are done based on BLOSUM50 matrix as it works well for recognizing very distant relationship. For protein sequence also, RPAlign SW can achieve a TG with respect to SW, and this is demonstrated by performing an all against all pairwise alignment for 100 protein sequences. Note that among the detected RPAs, there may be a few that are false positives. In order to determine the statistical significance of the detected RPAs, Karlin and Altschul statistics [39], [40] is implemented in the proposed paper. The $E()$ value is measured as follows:

$$E = KS_sS_l e^{-\lambda S} \tag{3}$$

where E is the expected frequency of chance occurrence of the RPA having similarity score S (or higher). K and λ are Karlin–Altschul parameters, where K is a small adjustment that takes into account the fact that optimal local alignment scores for alignments that start at different places in the two sequences may be highly correlated, and λ is a matrix-specific constant needed to convert a raw score into a bit score. Since there may be multiple nonoverlapping statistically significant RPAs, in this case, we utilize the Karlin–Altschule sum statistics to compute the final alignment score and $E()$ value [40]. The following equation describes the sum statistics:

$$S_{\text{sum}} = \lambda \sum_{i=1}^{r} S_i - \ln(KS_sS_l) - (r-1)(\ln(K)) + 2\ln(g) - \log(r!) \tag{4}$$

where S_i is the raw score of the ith RPA among r nonoverlapping RPAs and g is the size of the gaps between the RPAs. The sum score is then converted to sum probability (P-value) using the following equation:

$$P_r \approx \frac{e^{-S_{\text{sum}}} S_{\text{sum}}^{r-1}}{r!(r-1)!}. \tag{5}$$

RPAlign groups a set of RPAs only if the P-value is less than the P-value of any individual RPA. From among all such groups detected, the one having the lowest P_r-value is selected. Evidently, this set of RPAs provides the optimal alignable region, and finally, assigns the $E()$ value to a group of RPAs. For details, see [40].

The sensitivity and timing requirement of RPAlign SW, where the RPAs are selected as before, is compared with SW implementation in the following sections. In the same way, the sensitivity and timing requirement of RPAlign BLAST is also compared with BLAST.

TABLE III
SENSITIVITY COMPARISON OF DNA SEQUENCES BY SW, RPALIGN SW, BLAST, AND RPALIGN BLAST

				Sensitivity (%) with respect to SW																				
Pair	SW			RPAlign SW with θ/w= 0.85				RPAlign SW with θ/w= 0.9				RPAlign SW with θ/w= 0.95				BLAST				RPAlign BLAST with θ/w= .9				
	Score (Raw)	Score (Bits)	E ()	Score (Raw)	Score (Bits)	E ()	S (%)	Score (Raw)	Score (Bits)	E()	S (%)	Score (Raw)	Score (Bits)	E()	S (%)	Score (Raw)	Score (Bits)	E()	S (%)	Score (Raw)	Score (Bits)	E()	S(%) w.r.t. BLAST	S (%) w.r.t SW
P1	82	25.22	2.67e-02	82	25.22	2.67e-02	100	82	25.22	2.67e-02	100	82	25.22	2.67e-02	100	61	18.2	3.3	72.16	61	18.2	3.3	100	72.16
P2	89	27.16	2.88e-01	89	27.16	2.88e-01	100	73	22.73	6.21	83.69	73	22.73	6.21	83.69	UA	UA	UA	UA	UA	UA	UA	UA	UA
P3	108	32.42	3.79e-02	108	32.42	3.79e-02	100	96	29.10	3.79e-01	89.76	81	24.94	6.76	76.93	86	26.3	2.7	81.12	75	23.2	9.5e-02	88.21	71.56
P4	104	31.31	4.88e-01	104	31.31	4.88e-01	100	104	31.31	4.88e-01	100	89	27.16	8.70	86.74	101	30.4	9.1e-01	97.09	88	26.8	2.1e-01	88.16	85.59
P5	116	34.64	5.60e-02	116	34.64	5.60e-02	100	116	34.64	5.60e-02	100	96	29.10	2.61	84.01	116	34.64	5.9e-02	100	116	34.64	5.9e-02	100	100
P6	19582	5426.67	0.00	19582	5426.67	0.00	100	19582	5426.67	0	100	19582	5426.67	0	100	9153	2531	0	46.64	9153	2531	0	100	46.64
P7	1978	550.41	1.43e-156	1978	550.41	1.43e-156	100	1978	550.41	1.43e-156	100	1978	550.41	1.43e-156	100	1528	424	1e-118	77.03	1528	424	1e-118	100	77.03
P8	7563	2097.44	0	7563	2097.44	0	100	7563	2097.44	0	100	7563	2097.44	0	100	5698	1576	0	75.14	5698	1576	0	100	75.14

Input parameters: matrix = +5/–4; G/O = –25; G/E = –10; λ = 0.192; K = 0.176; H = 0.357.

TABLE IV
SENSITIVITY COMPARISON OF DNA SEQUENCES BY SW, RPALIGN SW, BLAST, AND RPALIGN BLAST

				Sensitivity (%) with respect to SW																				
Pair	SW			RPAlign SW with θ/w= 0.85				RPAlign SW with θ/w= 0.9				RPAlign SW with θ/w= 0.95				BLAST				RPAlign BLAST with θ/w= .9				
	Score (Raw)	Score (Bits)	E ()	Score (Raw)	Score (Bits)	E ()	S (%)	Score (Raw)	Score (Bits)	E()	S (%)	Score (Raw)	Score (Bits)	E()	S (%)	Score (Raw)	Score (Bits)	E()	S (%)	Score (Raw)	Score (Bits)	E()	S(%) w.r.t. BLAST	S (%) w.r.t
P1	91	27.71	4.74e-03	91	27.71	4.74e-03	100	91	27.71	4.74e-03	100	91	27.71	4.74e-03	100	62	17.7	4.5	63.87	62	17.7	4.5	100	63.87
P2	96	24.90	1.34	96	24.90	1.34	100	91	23.85	2.78	95.78	91	23.85	2.78	95.78	UA	UA	UA	UA	UA	UA	UA	UA	UA
P3	123	30.59	1.33e-01	123	30.59	1.33e-01	100	114	28.69	4.96e-01	93.79	103	26.38	2.47	86.24	UA	UA	UA	UA	UA	UA	UA	UA	UA
P4	149	36.06	1.80e-02	149	36.06	1.80e-02	100	124	30.80	6.93e-01	85.41	110	27.85	5.35e+00	77.23	UA	UA	UA	UA	UA	UA	UA	UA	UA
P5	138	33.75	1.03e-01	138	33.75	1.03e-01	100	138	33.75	1.03e-01	100	110	27.85	6.16	82.52	116	29.1	0.11	86.22	116	29.1	0.11	100	86.22
P6	21788	4593.96	0	21788	4593.96	0	100	21788	4593.96	0	100	21788	4593.96	0	100	15455	3260	0	70.96	15455	3260	0	100	70.96
P7	2599	552.12	4.36e-157	2599	552.12	4.36e-157	100	2599	552.12	4.36e-157	100	2599	552.12	4.36e-157	100	2152	457	1e-128	82.77	2152	457	1e-128	100	82.77
P8	10669	2251.93	0	10669	2251.93	0	100	10669	2251.93	0	100	10669	2251.93	0	100	8251	1742	0	77.35	8251	1742	0	100	77.35

Input parameters: matrix = +5/–4; G/O = –8; G/E = –6; λ = 0.146; K = 0.0390; H = 0.110.

TABLE V
TIME COMPARISON OF DNA SEQUENCES BY SW, RPALIGN SW, BLAST, AND RPALIGN BLAST

Time (Seconds)												
Pair	SW	RPAlign SW						BLAST		RPAlign BLAST		
		θ/w=0.85	TG (%)	θ/w=0.9	TG (%)	θ/w=0.95	TG (%)		TG (%)	θ/w= 0.9	TG (%) w.r.t. BLAST	TG (%) w.r.t. SW
P1	0.070	0.050	28.57	0.030	57.14	0.010	85.71	0.021	70	0.018	14.28	74.28
P2	2.710	0.440	83.76	0.390	85.61	0.160	94.10	UA	UA	UA	UA	UA
P3	14.260	9.320	34.64	3.060	78.54	0.310	97.83	0.103	99.28	0.069	33.01	99.52
P4	84.020	43.500	48.23	9.490	88.70	0.740	99.12	0.159	99.81	0.106	33.33	99.87
P5	95.350	46.889	50.82	10.520	88.97	0.960	99.99	0.177	99.81	0.086	51.41	99.91
P6	80.440	80.450	-1.24	80.450	-1.24	49.380	38.61	0.243	99.7	0.251	-3.29	99.69
P7	115.270	79.220	31.27	79.220	31.27	16.070	86.06	0.512	99.55	0.273	46.68	99.76
P8	4361.180	1682.760	61.41	1682.760	61.41	257.220	94.10	1.295	99.97	1.014	21.7	99.98

Input parameters: matrix = +5/–4; G/O = –25; G/E = –10; λ = 0.192; K = 0.176; H = 0.357.

A. Sensitivity Analysis

Table III shows a comparative study of SW with RPAlign SW, and BLAST with RPAlign BLAST with different values of θ/w in terms of alignment score (scores in bits) and the $E()$ values of the alignments. Here, the sensitivity (S) of RPAlign SW and RPAlign BLAST denotes the percentage of bit score with respect to those of SW and BLAST, respectively. According to Tables III and IV, irrespective of the effects of G/O and G/E, RPAlign SW with $\theta/w = 0.85$ always provides 100% sensitivity with respect to SW, whereas RPAlign SW with $\theta/w = 0.9$

TABLE VI
SENSITIVITY COMPARISON OF PROTEIN SEQUENCES BY SW, RPALIGN SW, AND BLAST USING BLOSUM62 AND BLOSUM50

Pair	Matrix	SW			RPAlign SW with θ/w= 4.5				RPAlign SW with θ/w= 4.6				RPAlign SW with θ/w= 4.7				BLAST			
		Score (Raw)	Score (Bits)	E ()	Score (Raw)	Score (Bits)	E ()	S (%)	Score (Raw)	Score (Bits)	E()	S (%)	Score (Raw)	Score (Bits)	E()	S (%)	Score (Raw)	Score (Bits)	E()	S (%)
R1	BL62	9893	4242.57	0	9893	4242.57	0	100	8295	3557.85	0	83.86	8295	3557.85	0	83.86	9893	4242.5	0	100
	BL50	12871	3366.3	0	12871	3366.3	0	100	12871	3366.3	0	100	12871	3366.3	0	100	12871	3366.3	0	100
R2	BL62	33068	14172.6	0	33068	14172.6	0	100	33068	14172.6	0	100	20971	8989.27	0	63.43	24347	10440	0	73.66
	BL50	42290	11048.4	0	42290	11048.42	0	100	42290	11048.4	0	100	42290	11048.4	0	100	40106	10480	0	94.85
R3	BL62	16222	6954.42	0	16222	6954.42	0	100	16222	6954.42	0	100	16222	6954.42	0	100	16222	6954.4	0	100
	BL50	20700	5410.67	0	20700	5410.67	0	100	20700	5410.67	0	100	20700	5410.67	0	100	20700	5410.6	0	100
R4	BL62	1204	519.5	6.49e-150	1204	519.5	6.49e-150	100	1204	519.5	6.49e-150	100	1204	519.5	6.49e-150	100	603	261	2e-72	50.24
	BL50	2653	698.09	1.07e-203	2653	698.09	1.07e-203	100	2653	698.09	1.07e-203	100	2653	698.09	1.07e-203	100	914	275	1e-76	39.39

Input parameters: For BLOSUM62, G/O = –11; G/E = –2; $\lambda = 0.297$; $K = 0.0820$; $H = 0.270$. For BLOSUM50, G/O = –12; G/E = –2; $\lambda = 0.181$; $K = 0.0250$; $H = 0.0950$.

TABLE VII
SENSITIVITY COMPARISON OF PROTEIN SEQUENCES BY SW, RPALIGN SW, AND BLAST

	BLOSUM50																		
Pair	SW			RPAlign SW with θ/w= 4.5				RPAlign SW with θ/w= 4.6				RPAlign SW with θ/w= 4.7				BLAST			
	Score (Raw)	Score (Bits)	E()	Score (Raw)	Score (Bits)	E()	S (%)	Score (Raw)	Score (Bits)	E()	S (%)	Score (Raw)	Score (Bits)	E()	S (%)	Score (Raw)	Score (Bits)	E()	S (%)
R5	37	14.98	2.29e-01	37	14.98	2.29e-01	100	37	14.98	2.29e-01	100	17	9.76	8.56	65.15	36	14.7	0.64	98.13
R6	41	16.0	1.23e-01	41	16.03	1.23e-01	100	41	16.03	1.23e-01	100	41	16.03	1.23e-01	100	41	16.0	0.12	100
R7	70	23.60	2.75	70	23.60	2.75	100	70	23.60	2.75	100	70	23.60	2.75	100	UA	UA	UA	UA
R8	75	24.9	9.82e-01	75	24.9	9.82e-01	100	75	24.9	9.82e-01	100	75	24.9	9.82e-01	100	75	24.9	.98	100
R9	74	24.65	1.09	74	24.65	1.09	100	67	22.82	3.88	92.58	67	22.82	3.88	92.58	63	21.8	8.1	88.44
R10	68	23.08	2.68	68	23.08	2.68	100	68	23.08	2.68	100	68	23.08	2.68	100	68	23.08	2.68	100

Input parameters: matrix = BLOSUM50; G/O = –12; G/E = –2; $\lambda = 0.181$; $K = 0.0250$; $H = 0.0950$.

provides the sensitivity range from 83.69% to 100% and 85.41% to 100%, respectively. To determine the statistical significance of the aligned regions detected by the RPAs, the $E(\,)$ value is computed. RPAlign SW with $\theta/w = 0.85$ always provides the same $E(\,)$ values as obtained from SW. Note that in most of the cases, $\theta/w = 0.9$ also provides 100% sensitivity with $E(\,)$ values similar to that of SW (except for P2 and P3 in Table III, and P2–P4 in Table IV, where both the sensitivities and $E(\,)$ values are decreased slightly). The same sequence pairs are also aligned by BLAST and RPAlign BLAST with $\theta/w = 0.9$ using similar parameters as mentioned in Tables III and IV. At the time of using G/O $= -25$ and G/E $= -10$, BLAST is unable to align P2 (see Table III), while the sensitivities for other pairs range from 46.64% to 100%, whereas in case of using G/O $= -8$ and G/E $= -6$, BLAST is unable to align P2–P4 (see Table IV), while the sensitivities for other pairs range from 63.87% to 86.22% with respect to SW. This clearly shows the superiority of RPAlign SW over BLAST and can become an alternate solution while SW is unable to align two large sequences. We have measured the sensitivities provided by RPAlign BLAST with $\theta/w = 0.9$, and except for P3 and P4 in Table III, RPAlign BLAST always provides 100% sensitivity with respect to BLAST. We have also measured the sensitivities provided by RPAlign SW with $\theta/w = 0.95$. From Tables III and IV, it is found that RPAlign SW with $\theta/w = 0.95$ provides sensitivities ranging from 76.93% to 100% and 77.23% to 100%, respectively, though in a few cases, the $E(\,)$ values are not satisfactory with respect to SW. But it still provides results superior to BLAST in most of the cases. Based on the previous observation, we recommend to use $\theta/w = 0.9$ at the time of running RPAlign SW or RPAlign BLAST as it provides the optimum sensitivity as compared with SW and BLAST, respectively.

Table VI shows the sensitivity comparisons of protein sequences by SW, RPAlign SW, and BLAST using two substitution matrices: BLOSUM62 and BLOSUM50. On the other hand, Table VII shows the sensitivity comparison of protein sequences that are distantly related in nature. From both the tables, it is clear that among the different values of θ/w ($\theta/w = 4.5$, 4.6, and 4.7), RPAlign SW provides a performance similar to that of SW using θ/w value around 4.5. The details of the TG obtained from each of the alignment is discussed in the following section.

B. Timing Analysis

Parallel implementation of the proposed method makes it efficient in terms of speedup of the computation. The timing analysis is provided here based on the same set of DNA sequences considered earlier. According to the timing results in Table V, as expected, RPAlign BLAST requires the lowest computation time followed by that for BLAST. Again SW always requires the largest computation time, while RPAlign SW with different values of θ/w, in general, provides an improvement over the SW time. Among the latter, RPAlign SW with $\theta/w = 0.95$ provides the largest speedup. P6 presents an interesting case where no TG is observed for $\theta/w = 0.85$ and 0.9 since the sequences are very similar. In fact, here the alignment time is the same as that for SW. The additional RPA detection time results in an overall negative TG. Moreover, it becomes evident from the result of P6 that the detection time of RPAs is only a very small factor of the alignment time. As can be seen from Table V, the TGs obtained by RPAlign SW with respect to SW (except for P6)

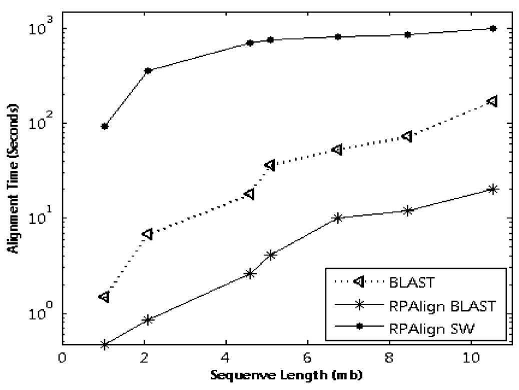

Fig. 4. Speed comparison among RPAlign SW, RPAlign BLAST, and BLAST.

are 28.57%–83.76%, 31.27%–88.97%, and 85.71%–99.99% for the different values of $\theta/w = 0.85$, 0.9, and 0.95, respectively. Table V also shows that BLAST achieves a TG of 70%–99.97% as compared to SW. RPAlign BLAST further improves upon the time of BLAST (except P6) while keeping the same quality of alignment in almost all the cases. Fig. 4 shows the speed comparison among RPAlign SW, BLAST, and RPAlign BLAST for seven pairs of magabase-scale DNA sequences. For these sequence pairs, SW is unable to perform the alignment because of the sequence sizes. The figure shows that RPAlign BLAST efficiently enhances the performance of BLAST by a significant margin. The computation time for RPAlign SW, though greater than that for BLAST, is still manageable. Interestingly, it was observed that BLAST shows a tendency to come closer. This presents an interesting application of the proposed method that can be used in conjunction with SW for megabase-scale sequences with high sensitivity, as characteristic of SW, but with significantly reduced time requirement. We observed that for the protein sequences also, RPAlign SW can achieve ~40% TG with respect to SW. To demonstrate this fact, in a part of the investigation, an attempt was made to perform an all against all pairwise alignment for 100 protein sequences (average length >4 kb), and we obtained a significant TG without losing much sensitivity. For 100 sequences, a total of 4950 comparisons are performed. While SW implementation needs 47 min and 32 s, RPAlign SW takes 29 min and 03 s, i.e., the TG is obtained by RPAlign SW is 38.88%. BLAST has taken much lesser time, which is only 5 min and 56 s, but at the cost of much reduced sensitivity, and it was also unable to align in some cases.

V. Discussion and Conclusion

An MPI-based parallel algorithm for performing pairwise local alignment through the detection of RPAs has been proposed. It has been observed that the proposed method can provide an alignment quality comparable to that of the SW algorithm while requiring significantly less time. Although BLAST has reduced the runtime compared with the best known SW implementation, it has significantly low sensitivity, particularly for the sequences that are distantly related, and thus, does not reflect the actual biological evidence. RPAlign is not a new alignment algorithm; rather it can be used with any existing pairwise alignment technique to enhance its performance. RPAlign SW can efficiently detect multiple statistically significant regions in parallel. This feature is essential for finding all exons in a multiexon gene sequence, all complete or partial copies of a repetitive element in a genomic sequence or multiple domains of a protein sequence. In contrast, the original SW implementation can show only one optimal alignment region. If the proposed algorithm is used with BLAST, it runs much faster with the same quality of output. If it is used with SW implementation, then the required time is much lesser and comparable to that of SW. The RPAlign algorithm thus allows the researchers to obtain SW-like sensitivity, while requiring significantly less time. Our aim is to enhance the existing methodologies in terms of speed without losing the sensitivity provided by them by utilizing the power of parallel processing. It can efficiently align not only the large DNA sequences but also the more complex protein sequences, and even the sequences that are distantly related. The efficiency of the proposed method derives from the fact that it is able to appropriately prune those regions of the sequence pair that will not take part in the final alignment (which the SW algorithm unnecessarily tries to align). This, in turn, results from the detection of the regions where alignment is possible. In the case of distantly related sequence pairs, the gain is much more, since, in these cases, a large amount of effective pruning is possible. As can be seen from Tables III and IV, for P7 and P8, RPAlign SW always provides 100% sensitivity while gaining the time by increasing the value of θ/w. The reason behind is that the alignable region of the two sequences is comparatively very small with respect to the length of the input sequences. But SW also needs to consider unalignable regions to compute the scoring matrix. Thus, computation time is increased unnecessarily. For example, in case of P7 in Table III, the actual alignable region of the two input sequences detected by the SW are only 3% and 0.82% with respect to the input sequence length. But to align this region, SW needs to consider the entire length of the two sequences. On the other hand, RPAlign SW with $\theta/w = 0.85$ and 0.9 prunes the unalignable regions 0% and 71.5% from the two input sequences, while RPAlign SW with $\theta/w = 0.95$ prunes 70.5% and 91.7%, respectively. Thus, increased TG is obtained while acquired sensitivity remains the same. RPAlign SW with $\theta/w = 0.95$ is upto 99 times faster than SW, but loses some sensitivities in few experiments, whereas RPAlign SW with $\theta/w = 0.9$ is upto 9 times faster, but provides sensitivity similar to that of SW. For megabase-scale sequences, RPAlign BLAST is found to be three to nine times faster than BLAST. As RPAlign produces high quality of alignment in a significantly lesser time, we are currently investigating new multiple sequence alignment techniques in a parallel framework. Extending the work to multiple sequence alignment is not trivial, and will constitute an important area of future research.

Acknowledgment

The authors thank Dr. S. Sur-Kolay, Associate Professor, Advanced Computation and MicroElectronics Unit (ACMU), Indian Statistical Institute, Kolkata, India, for providing access to the cluster computer.

REFERENCES

[1] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, B. A. Rapp, and D. L. Wheeler, "GenBank," *Nucleic Acids Res.*, vol. 28, pp. 15–18, 2000.

[2] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, pp. 195–197, 1981.

[3] O. Gotoh, "An improved algorithm for matching biological sequences," *J. Mol. Biol.*, vol. 162, pp. 705–708, 1982.

[4] E. W. Myers, "An *O(ND)* difference algorithm and its variations," *Algorithmica*, vol. 1, pp. 251–266, 1986.

[5] E. W. Myers and W. Miller, "Optimal alignments in linear space," *Bioinf. Oxford J.*, vol. 4, pp. 11–17, 1988.

[6] S. Batzoglou, L. Pachter, J. P. Mesirov, B. Berger, and E. S. Lander, "Human and mouse gene structure: Comparative analysis and application to exon prediction," *Genome Res*, vol. 10, pp. 950–958, 2000.

[7] M. Brudno, C. B. Do, G. M. Cooper, M. F. Kim, E. Davydov; NISC Comparative Sequencing Program, E. D. Green, A. Sidow, and S. Batzoglou, "LAGAN and multi-LAGAN: Efficient tools for large-scale multiple alignment of genomic DNA," *Genome Res.*, vol. 13, pp. 721–731, 2003.

[8] W. Pearson and D. Lipman, "Improved tools for biological sequence comparison," *Proc. Nat. Acad. Sci. USA*, vol. 85, pp. 2444–2488, 1988.

[9] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *J. Mol. Biol.*, vol. 215, pp. 403–410, 1990.

[10] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," *Nucleic Acids Res*, vol. 25, pp. 3389–3402, 1997.

[11] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller, "A greedy algorithm for aligning DNA sequences," *J. Comput. Biol.*, vol. 7, pp. 203–214, 2000.

[12] T. A. Tatusova and T. L. Madden, "BLAST 2 SEQUENCES, A new tool for comparing protein and nucleotide sequences," *FEMS Microbiol. Lett.*, vol. 174, pp. 247–250, 1999.

[13] W. Gish. (1995). WU-BLAST [Online]. Available: http://blast.wustl.edu/

[14] S. Schwartz, Z. Zhang, K. A. Frazer, A. Smit, C. Riemer, J. Bouck, R. Gibbs, R. Hardison, and W. Miller, "PipMaker—A web server for aligning two genomic DNA sequences," *Genome Res.*, vol. 10, pp. 577–586, 2000.

[15] B. Ma, J. Tromp, and M. Li, "Pattern Hunter: Faster and more sensitive homology search," *Bioinformatics*, vol. 18, pp. 440–445, 2002.

[16] W. J. Kent, "BLAT—The BLAST-like alignment tool," *Genome Res.*, vol. 12, pp. 656–664, 2002.

[17] Z. Ning, A. J. Cox, and J. C. Mullikin, "SSAHA: A fast search method for large DNA databases," *Genome Res*, vol. 11, pp. 1725–1729, 2001.

[18] T. Kahveci, V. Ljosa, and A. K. Singh, "Speeding up whole-genome alignment by indexing frequency vectors," *Bioinformatics*, vol. 20, pp. 2122–2134, 2004.

[19] K. R. Rasmussen, J. Stoye, and E. W. Myers, "Efficient *q*-gram filters for finding all epsilon-matches over a given length," in *Proc. 9th Conf. Comput. Mol. Biol. (RECOMB)*, Cambridge, MA, 2005, pp. 189–203.

[20] K. J. Kalafus, A. R. Jackson, and A. Milosavljevic, "Pash: Efficient genome-scale sequence anchoring by positional hashing," *Genome Res.*, vol. 14, pp. 672–678, 2004.

[21] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg, "Alignment of whole genomes," *Nucleic Acids Res.*, vol. 27, pp. 2369–2376, 1999.

[22] S. Burkhardt, A. Crauser, P. Ferragina, H. P. Lenhof, E. Rivals, and M. Vingron, "Q-gram based database searching using a suffix array (QUASAR)," in *Proc. Annu. Int. Conf. Comput. Mol. Biol. (RECOMB)*, ACM Press, France, 1999, pp. 77–83.

[23] S. Kurtz and C. Schleiermacher, "REPuter: Fast computation of maximal repeats in complete genomes," *Bioinformatics*, vol. 15, pp. 426–427, 1999.

[24] N. Bray, I. Dubchak, and L. Pachter, "AVID: A global alignment program," *Genome Res.*, vol. 13, pp. 97–102, 2003.

[25] R. Braun, K. T. Pedretti, T. L. Casavant, T. E. Scheetz, C. L. Birkett, and C. A. Roberts, "Parallelization of local blast service on workstation clusters," *Future Gen. Comput. Syst.*, vol. 17, pp. 745–754, 2001.

[26] N. Camp, H. Cofer, and R. Gomperts. (1998). High-throughput blast [Online]. Available: http://www.sgi.com/industries/sciences/chembio/resources/papers/HTBlast/HTWhitepaper.html

[27] E. H. Chi, E. Shoop, J. Carlis, E. Retzel, and J. Riedl, "Efficiency of shared memory multiprocessors for a genetic sequence similarity search algorithm," Comput. Sci. Dept., Univ. Minnesota, Crookston, Tech. Rep. TR97-05, 1997.

[28] R. D. Bjornson, A. H. Sherman, S. B. Weston, N. Willard, and J. Wing, "Turboblast (r): A parallel implementation of blast built on TurboHub," in *Proc. Int. Parallel Distrib. Process. Symp.*, 2002, p. 325.

[29] D. Mathog, "Parallel blast on split databases," *Bioinformatics*, vol. 19, pp. 1865–1866, 2003.

[30] A. E. Darling, L. Carey, and W. Feng, "The design, implementation, and evaluation of mpi-BLAST," presented at the Cluster World Conf. Expo, in Conjunction with 4th Int. Conf. Linux Clusters: The HPC Revolution, San Jose, CA, 2003.

[31] H. Lin, X. Ma, P. Chandramohan, A. Geist, and N. Sarnatova, "Efficient data access for parallel blast," in *Proc. Int. Parallel Distrib. Process. Symp.*, 2005, p. 72b.

[32] T. Rognes, "ParAlign: A parallel sequence alignment algorithm for rapid and sensitive database searches," *Nucleic Acid Res.*, vol. 29, pp. 1647–1652, 2001.

[33] E. C. Osório, J. E. de Souza, A. C. Zaiats, P. S. L. de Oliveira, and S. J. De Souza, "pp-Blast: A pseudo-parallel Blast," *Braz. J. Med. Biol. Res.*, vol. 36, pp. 463–464, 2003.

[34] C. Oehmen and J. Nieplocha, "ScalaBLAST: A scalable implementation of BLAST for high-performance data-intensive bioinformatics analysis," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 8, pp. 740–749, Aug. 2006.

[35] T. Rognes and E. Seeberg, "Six-fold speed-up of Smith–Waterman sequence database searches using parallel processing on common microprocessors," *Bioinformatics*, vol. 16, pp. 699–706, 2000.

[36] M. Farrar, "Striped Smith–Waterman speeds database searches six times over other SIMD implementations," *Bioinformatics*, vol. 23, pp. 156–161, 2007.

[37] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proc. Nat. Acad. Sci. USA*, vol. 89, pp. 10915–10919, 1992.

[38] J. M. Chandonia, G. Hon, N. S. Walker, L. L. Conte, P. Koehl, M. Levitt, and S. E. Brenner, "The ASTRAL compendium in 2004," *Nucleic Acids Res.*, vol. 32, pp. D189–D192, 2004 (database issue).

[39] S. Karlin and S. F. Altschul, "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes," *Proc. Nat. Acad. Sci. USA*, vol. 87, pp. 2264–2268, 1990.

[40] S. Karlin and S. F. Altschul, "Applications and statistics for multiple high-scoring segments in molecular sequences," *Proc. Nat. Acad. Sci. USA*, vol. 90, pp. 5873–5877, 1993.

Sanghamitra Bandyopadhyay (M'99–SM'05) received the M.Tech. degree from the Indian Institute of Technology (IIT) Kharagpur, Kharagpur, India, and the Ph.D. degree from the Indian Statistical Institute (ISI), Kolkata, India, both in computer science.

She has worked at the Los Alamos National Laboratory, USA; the University of New South Wales, Sydney, Australia; the University of Texas at Arlington, USA; the University of Maryland at Baltimore, USA; Fraunhofer Institute, Germany; Tsinghua University, China; and the University of Rome. She is a Professor at ISI. She has authored or coauthored more than 150 technical articles published in journals, book chapters, and conference proceedings, and authored/edited three books. Her current research interests include computational biology and bioinformatics, soft and evolutionary computation, and pattern recognition and data mining.

Dr. Bandyopadhyay was the first recipient of the Dr. Shanker Dayal Sharma Gold Medal and also the Institute Silver Medal for the best all-around post-graduate performer at IIT Kharagpur in 1994. She has also received the Young Scientist/Engineer Awards of the Indian National Science Academy, the Indian Science Congress Association, and the Indian National Academy of Engineers. She was also awarded the Swarnajayanti Fellowship by the Department of Science and Technology, Government of India.

Ramkrishna Mitra received the M.Sc. degree in bioinformatics from Sikkim Manipal University of Health, Medical and Technological Sciences, Manipal, India.

He is currently a Junior Research Fellow in a DST-sponsored project at the Machine Intelligence Unit, Indian Statistical Institute, Kolkata, India. His current research interests include parallel processing, bioinformatics, and data mining.