Bioinformatics Advance Access published October 6, 2007

Sequence analysis

Biclustering As A Method For RNA Local Multiple Sequence Alignment

Shu Wang[1]*, Robin R. Gutell[2] and Daniel P. Miranker[3]

[1]Department of Electrical and Computer Engineering, [2] School of Biological Sciences, Section of Integrative Biology, and [3]Department of Computer Science, University of Texas At Austin, Austin, TX 78712, USA

Associate Editor: Prof. Keith Crandall

ABSTRACT

Motivations: Biclustering is a clustering method that simultaneously clusters both the domain and range of a relation. A challenge in multiple sequence alignment (MSA) is that the alignment of sequences is often intended to reveal groups of conserved functional subsequences. Simultaneously, the grouping of the sequences can impact the alignment; precisely the kind of dual situation biclustering is intended to address.

Results: We define a representation of the MSA problem enabling the application of biclustering algorithms. We develop a computer program for local MSA, BlockMSA, that combines biclustering with divide-and-conquer. BlockMSA simultaneously finds groups of similar sequences and locally aligns subsequences within them. Further alignment is accomplished by dividing both the set of sequences and their contents. The net result is both a multiple sequence alignment and a hierarchical clustering of the sequences.

BlockMSA was tested on the subsets of the BRAliBase 2.1 benchmark suite that display high variability and on an extension to that suite to larger problem sizes. Also, alignments were evaluated of two large data sets of current biological interest, T box sequences and Group IC1 Introns. The results were compared with alignments computed by ClustalW, MAFFT, MUCLE, and PROBCONS alignment programs using Sum of Pairs (SPS), and Consensus Count.

Results for the benchmark suite are sensitive to problem size. On problems of 15 or greater sequences, BlockMSA is consistently the best. On none of the problems in the test suite are there appreciable differences in scores among BlockMSA, MAFFT and PROBCONS. On the T box sequences, BlockMSA does the most faithful job of reproducing known annotations. MAFFT and PROBCONS do not. On the Intron sequences, BlockMSA, MAFFT and MUSCLE are comparable at identifying conserved regions.

Availability: BlockMSA is implemented in Java. Source code and supplementary data sets are available at http://aug.csres.utexas.edu/msa/.

Contact: swang5@ece.utexas.edu

1 INTRODUCTION

Clustering, as it is commonly used in Bioinformatics, takes input arranged in a data matrix, where the rows correspond to data objects and the columns correspond to their features/attributes. Biclustering clusters both rows and columns simultaneously (Madeira and Oliveira, 2000; Dhillon, 2001, Barkow *et. al.,* 2006). In biclustering, each object in a cluster is selected only using a subset of features and each feature in a cluster is selected only using a subset of objects. In this way, it discovers local signals/coherences in a data matrix, and derives local clusters within the data matrix.

A challenge in multiple sequence alignment (MSA) is that sets of sequences usually contain unknown subsets representing phylogenetic or functional groups (Notredame, 2002). While aligning such a set of sequences is intended to reveal such groupings, the sequence alignment itself is the result of the grouping; precisely the chicken-and-egg scenario biclustering is intended to address.

In order to apply biclustering to the MSA problem, we define the following mapping from MSA to biclustering (Fig.1). Specifically, we represent the MSA problem in a biclustering matrix. Given a set of sequences, we first identify candidate "blocks", possible local alignments of multiple subsequences. We then use them to construct a biclustering matrix where each row corresponds to a candidate block and each column corresponds to a sequence. The value in the matrix is "1" if a block is on the sequence, "0", otherwise. Biclustering is a two-way clustering. Instead of clustering sequences over all blocks, biclustering can cluster sequences with respect to subsets of blocks and vice versa (Fig.2). For each identified bicluster matrix, its columns consist of a subset of sequences, corresponding to a sequence group and its rows consist of a subset of blocks, corresponding to the conserved features for the sequence group (Fig.3).

We have developed a program, BlockMSA, that exploits biclustering to align non-coding RNA (ncRNA) datasets. We recursively apply biclustering by excluding the aligned blocks from further considerations and continue the MSA in a divide-and-conquer fashion, one sub-problem for each sequence group.

ncRNA genes produce some of the cell's most important products, including transfer RNA (tRNA) and ribosomal RNA (rRNA) (Cannone *et. al* ., 2002;Griffiths-Jones *et. al.*, 2005). However, the role of ncRNA has been underestimated for a long time. These RNAs are now being implicated in various regulatory functions, in addition to their roles in protein synthesis (Economist, 2007). As a result of this recent change, the number of ncRNAs and our understanding of their importance in cellular metabolism will increase dramatically in the next few years.

*To whom correspondence should be addressed: Shu Wang, e-mail: swang5@ece.utexas.edu

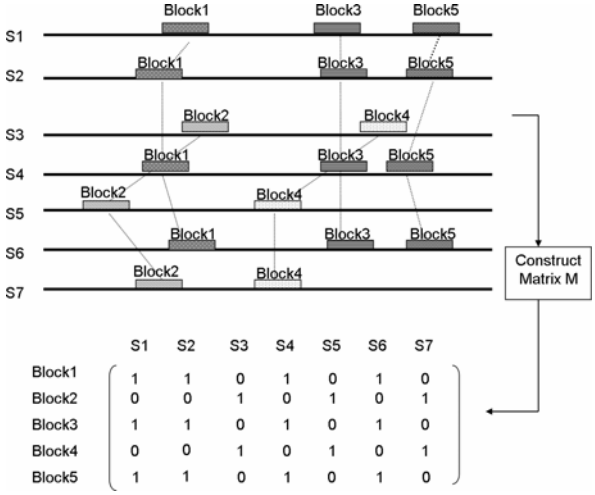

Fig. 1. MSA To Biclustering Mapping. Given a set of sequences, {*S1, S2, …, S7*}, we first identify the possible "blocks", local multiple alignments of subsequences. We then represent the MSA problem in a biclustering matrix *M*, where each row corresponds to a block and each column corresponds to a sequence. The value in the matrix is "1" if a block is on the sequence, "0", otherwise.

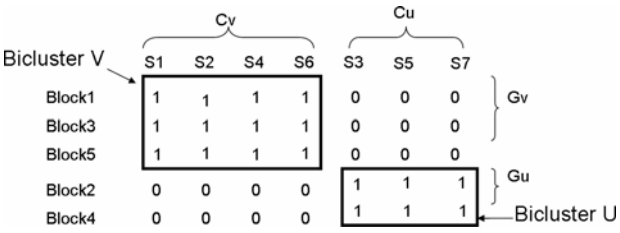

Fig. 2. Biclustering Matrix *M*. To bicluster the matrix *M*, we applied the biclustering program, BiMax, to it. BiMax rearranged the rows and columns of Matrix M and identified two biclusters *U* and *V*. In this example, a bicluster is defined as a sub-matrix with all its elements equal to 1.

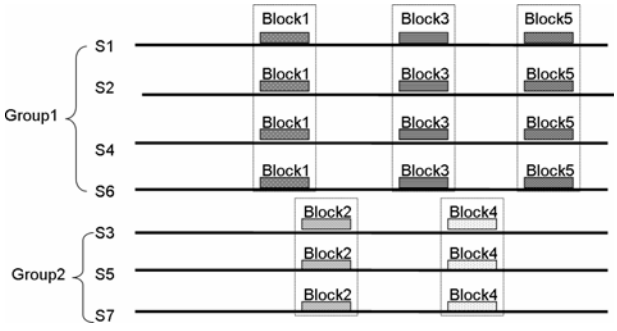

Fig. 3. Map Biclustering results back to MSA. For each bicluster (*U* or *V*), the columns (sequences) correspond to a sequence group and the rows (blocks) represent the local alignments in the sequence group. Bicluster V consists of the sequence group $\{S_1, S_2, S_4, S_6\}$ and its local alignments are $\{Block_1, Block_3, and\ Block_5\}$. Bicluster *U* consists of the sequences $\{S_3, S_5, S_7\}$ and its local alignments on them are $\{Block_2, Block_4\}$.

Improving the speed, accuracy and scalability of MSA is becoming an increasingly important task. Large scale MSA is especially needed for highly variable ncRNA families to reveal their functionally conserved regions. Experience is showing that the larger the set of sequences considered, the more biological details reliably emerge. For example, comparative analysis of multiple RNA sequence alignments has revealed new types of base pairings (e.g. U:U and C:C instead of the canonical A:U, G:C, and G:U) and new structural motifs (Gutell *et. al.*, 1994). Current global MSA methods, such as ClustalW, are useful for closely related RNAs (Thompson *et. al.* 1994). They become less effective when the number of sequences increases and the sequences are more variable. Current local MSA methods such as Dialign 2 also become less effective for large and variable sequence alignment problems (Morgenstern, 1999).

We applied BlockMSA to highly variable ncRNA for local MSA. We tested BlockMSA on three sources of problems and compared its results with alignments computed by ClustalW (Thompson *et. al.*, 1994**)**, MAFFT (Katoh *et. al.*, 2005), MUSCLE (Edgar, 2004) and PROBCONS (Do *et. al.*, 2005). Since the intention is to support the analysis of large sets of highly variable ncRNAs, first we determined the subsets of BRAliBase 2.1 (Wilm *et. al.*, 2006) that display high variability. Second, since the maximum problem size in BRAliBase 2.1 is 15 sequences, we increased test set size and formed problem instances as large as 80 sequences. Third, two large data sets of current biological interest, from the Comparative RNA Web (CRW) Site, are evaluated, T box sequences (Grundy & Henkin, 2003) and Group IC1 Introns **(**Cannone et. al., 2002).

We found on the large-scale benchmark based testing that the alignment programs score in a consistent fashion. BlocksMSA consistently scores best for larger problems, 15 sequences or greater. Otherwise, BlockMSAs performance scores competitively with the best scores produced by the other alignment programs.

Results on the two large biological tests demonstrate that BlocksMSA is the most effective. This is only to be expected as BlocksMSA was explicitly developed for large-scale problems. What was unexpected is that the formal measure of alignment quality, SPS, does not always bear correlation with biological effectiveness.

1.1 Review of Current MSA

The combinatorial complexity of MSA makes optimal solutions infeasible. Thus, popular MSA programs are heuristic. MSA programs can be grouped into sequence or block based methods.

The main sequence-based method is progressive alignment (e.g. ClustalW). Progressive methods hierarchically align the sequences/sequence groups by adding one sequence or sequence group at a time (Thompson *et. al.*, 1994). Progressive methods are fast. However, errors made in early alignment stages are propagated and accumulated into later stages. Iterative refinement methods seek to overcome this problem. An iterative refinement algorithm first produces an initial alignment and then refines it through a series of iterations to correct errors made in progressive steps. MAFFT, MUSCLE and PROBCONS are iterative refinement methods. Iterative methods still have difficulty when sequences are highly variable. This is because conserved regions may occur within long unaligned regions. Such regions can often only be identified by a broad global search. Iterative methods tend to make local decisions.

Block-based MSA methods directly look for the vertically conserved regions among sequences. Block-based methods can be further characterized as consistency or probability-based. Dialign 2 is representative of consistency-based methods. It is a greedy method that first identifies conserved regions (blocks) shared by two sequences, and then adds blocks to construct the final alignment based on their consistency weights. Dialign 2's time complexity, $O(n^4l^2)$ for n sequences of length l, prevents it from scaling to large problems. Probability-based methods, such as MEME (Bailey and Elkan, 1994) and Gibbs Sampling (Lawrence *et. al.*, 1993), start from an initial random choice of an alignment then recursively realign the sequences. The process repeats until the system converges. Current formulations require conserved regions to be present in all the sequences. They need "prior" assumptions on the number of blocks, which is rarely known a priori. Like iterative refinement methods, probabilistic methods tend to be local in nature. Probabilistic methods are computation intensive making them a difficult starting point for scalable algorithms.

2 SYSTEMS AND METHODS

2.1 Alignment Test Sets

Currently, there is no standard benchmark for RNA local MSA. The BRAliBase 2.1 benchmark suite is not specifically designed for local alignment testing. It has the datasets with percentage identity ranging from 20% to 95% and the maximum number of sequences per test set is 15, which is not enough for testing scalability and robustness. Further, the ratio between the average sequence length and the total number of columns in a reference alignment is usually high, which does not represent cases with large insertion/deletions.

Based on the analysis above, we chose three types of test sets: (1) The subsets of BRAliBase which are highly variable and suitable for local MSA; (2) LocalExtR, an extension of BRAliBase 2.1, comprising larger-scale test groups and patterned on BRAliBase 2.1. (3) LSet, a pair of large-scale test sets representative of current biological problems.

The subsets from BRAliBase are selected from the most variable test sets within the suite. They are from the THI, Glycine riboswitch and Yybp-Ykoy RNA families, and contain 232 test datasets.

LocalExtR uses the same seed alignments from Rfam that BRAliBase uses and forms larger test groups. The BRAliBase 2.1 convention is to label a test group *ki*, where *i* is the number of sequences for each test set in the group. We created four new test groups, *k20*, *k40*, *k60* and *k80*, totaling 90 test sets. The new test sets maintain the percentage identity to be less than 60%. To model large insertions/deletions, the ratio of the average sequence length to the total number of columns in the reference alignments is as small as 0.36 (See Table 1)

LSet contains a set of 248 T box leader sequences and a set of 90 Group IC1 Introns from the CRW Site. T box leader sequences are located upstream of many aminoacyl-tRNA synthetase (AARS), amino acid biosynthesis and amino acid transporter genes in gram-positive bacteria. Group IC1 Introns represent a family of RNA molecules with a specific higher-order structure and the ability to catalyze their own excision by a common splicing mechanism (Cech, 1990). In addition to containing many more sequences, both the average sequence length and the differences in sequence lengths are much larger than for the BRAliBase test sets (See Table 2).

Table 1. Measures of the BRAliBase and LocalExtR Test Groups

Test Group		# of data sets	Sequence Length, Average Of Each Value Over ki Set			Average Ratio: Avg-Length/ Reference	Avg PI (%)
			Avg.	Min.	Max.		
BRAliBase 2.1 Subsets 232 datasets	k5	124	109	96	125	0.79	51.1
	k7	62	110	94	131	0.75	49.8
	k10	32	108	94	129	0.72	49.3
	k15	14	110	88	137	0.67	49.3
LocalExtR 90 datasets	k20	30	115	90	172	0.53	48.4
	k40	25	114	87	180	0.47	48.5
	k60	20	107	81	189	0.40	50.7
	k80	15	106	77	204	0.36	54.8

k*i* indicates a test group containing *i*-sequence datasets. Note there are many test datasets for each *ki*, (See Supplementary Data). The table details, the number of test datasets, the average sequence length, and the average of the minimum and maximum sequence length per test sets within a group. Similarly, the ratio of the average sequence length to the reference alignment's length and percentage identity (PI).

Table 2. LSet.

Test Group	Data-Set Size	Sequence Length			Ratio Avg-Length/ Reference	PI (%)
		Avg.	Min	Max		
T box	248	269	78	365	0.40	40.0
Group IC1 Intron	90	563	347	1910	0.12	38.0

The table details the number of sequences in a test dataset, average, minimum and maximum sequence length, the ratio of the average sequence length to the reference alignment's length and the percentage identity (PI).

2.2 Scoring Alignments

We used two independent yet complementary scores to evaluate alignment quality, the Sum-of-Pairs Score (SPS) and Consensus Count.

SPS (Thompson *et. al.*, 1999) measures the level of sequence consistency between a test and a reference alignment by comparing all possible pairs per column between both alignments; it ranges from 0 (no agreement) to 1 (complete agreement).

The Consensus Count is a measurement for column conservation. It can be computed without identifying a reference alignment. Given a threshold, a consensus sequence represents each column of an alignment with the majority character in that column. Specifically the major character's percentage in the column should be greater or equal to the given threshold. The consensus sequence represents a column with a gap if the major character's percentage in that column is less than the specified threshold. After we obtain the consensus sequence in an alignment, the Consensus Count is the number of non-gap residues within the consensus sequence. It measures the vertical conservation in an alignment.

2.3 Biological Validation

In order to test the biological effectiveness of the different alignments we compared the output of the five programs (Block-MSA, ClustalW, MAFFT, MUSCLE, and PROBCONS) to each other and noted their ability to correctly align conserved areas. The T box sequences have been studied enough that we could make a quantifiable assessment with respect to known conserved functional subsequences. First, all the T box gene sequences have evolved from a common ancestor and contain a conserved 14-nucleotide sequence, 5•-AGGGUGGNACCGCG-3• (Grundy and Henkin 2003). In addition, the T box dataset contains sequences from 12 major gene groups. Each gene group shares a common triplet sequence representing a specifier sequence codon for the amino acid matching the amino acid class of the downsteam coding region (Grundy and Henkin 2003). For example, all tyrosyl genes contain a UAC tyrosine codon, leucine genes contain a CUC leucine codon. Identification of the specifier sequence in each gene group can provide insights into amino acid specificity. Here a major gene group in the reference alignment is defined as having more than four sequences and strongly showing a specifier sequence pattern. This is because if a gene group does not have enough representative sequences, the specifier sequence feature may not be apparent. Thus, consistent with the biological goals of a local multiple RNA sequence alignment, the T box test set allows us to check if a program correctly align the T box motif and count the number of specifier sequences successfully identified by a program.

The Intron data set is not yet as well annotated as the T box data set. We simply note the program's relative ability to identify conserved regions.

3. ALGORITHM

The algorithm has three main steps.

1) Identify candidate blocks.
2) Represent MSA as a biclustering problem. Apply biclustering to simultaneously cluster sequences into groups and find the conserved regions within each group.
3) For each sequence group, recursively apply the above two steps, until the sequence group is small enough or no more conserved regions are found within it.

We first provide key definitions and then expand on each of the three steps.

Definition 1. A *k*-block (conserved region) represents an un-gapped alignment region in *k* sequences, consisting of equal-length fragments from each of *k* sequences respectively, $B=\{f_1, f_2,\ldots, f_k\}$, where f_i represents a fragment from sequence *i*.

Definition 2. A *2*-block is a special case of *k*-block with *k=2*, which represents an un-gapped alignment region in a pairwise alignment.

Definition 3. Block Similarity Score. Given a *k*-block $B=\{f_1, f_2, \ldots, f_k\}$, the score of B is the sum of all the similarity scores of the $\binom{k}{2}$ pairs of fragments(*2*-blocks) within B:

$$S(B)=\sum_{1\le i<j\le k} S(f_i, f_j)$$

where $S(f_i, f_j)$ is the similarity score of fragments f_i and f_j.

Definition 4. **Fragment Order.** Given two fragments *a* and *b* on a sequence, $a=(a_1,\ldots,a_{k1})$ and $b=(b_1,\ldots,b_{k2})$, where a_i and b_i represent two residues from the same sequence, *a* is said to be less than *b* (written as $a<b$) if and only if the ending position of *a* is less than the beginning position of *b*.

Definition 5. **Block Order**. Given two *k*-blocks on *k* sequences, $F=(f_1, f_2,..,f_k)$ and $G=(g_1,g_2,...,g_k)$, where f_i and g_i represent two fragments from the same sequence *i*, Block *F* is less than Block *G* (written as $F < G$) if and only if $\forall i \in [1, k],\ f_i<g_i$.

Definition 6. **Non-overlapping Blocks.** Given two *k*-blocks *F* and *G* on *k* sequences, they are non-overlapping if and only if *F* is less than *G* or *G* is less than *F*.

Definition 7. Chain. A set of *k*-blocks $B=\{b_1,b_2,..,b_n\}$ on *k* sequences is called a chain if all the blocks in *B* are non-overlapped. The score of a chain is the sum of the scores of all its blocks minus the gap costs between them.

$$Score_{chain}(B) = [\sum_{1\le i<n} S(b_i) - Gap(b_i, b_{i+1})] + S(b_n)$$

Definition 8. An Optimal Set of Blocks. Given a set of blocks *B*, an optimal set of blocks is the chain with maximum score over all the chains of *B*.

3.1 Identify Candidate Blocks

Typically a given set of RNA sequences has multiple conserved-regions within it. Our goal here is to identify the set of possible conserved regions. Our algorithm is heuristic, consisting of three main steps: (1) dividing the sequence into overlapping k-length fragments, (2) calculation of fragment similarity scores, and (3) construction of candidate blocks. Each of the above steps is detailed as follows:

Step (1) Dividing sequences into fragments

For each sequence, it is divided into overlapping *k*-length fragments.

Step (2) Calculation of fragments' similarity scores

We first generate a library of pairwise alignments for all possible sequence pairs. The pairwise alignments can be obtained from any external sequence alignment program. We used ClustalW to do the pairwise alignment.

For each pairwise alignment, a window of length *k* is slid through it and each position of the window produces a *k*-length sub-region. We only keep those un-gapped sub-regions whose similarity scores are greater than a pre-specified threshold. Thus each sub-region is actually a *2*-block. Hence, for each *2*-block, its un-gapped alignment region provides its fragments' similarity score. For the fragments not appearing in the alignments, their similarity score is set to 0.

Step (3) Construction of candidate blocks

In this step, we construct block candidates from the pool of *2*-blocks. To do so, we use a greedy clustering-based method. We can think of each fragment as a point and an *n*-block as an *n*-point cluster. To determine an *n*-point cluster, we start the initial cluster with a seed point (fragment) and gradually add one fragment from each sequence to the cluster. Hence an *n*-point cluster will consist of *n* fragments from *n* sequences and form an *n*-block. To add a fragment to the cluster, we pick the fragment which has the maximum sum of similarity scores with the fragments in the cluster. The reason behind this is if a fragment has a higher similarity score with a cluster, it means it has more alignment consistency with the fragments in the cluster and shows more vertical conservation consistency. By using the clustering method to construct blocks, we directly integrate consistency calculation into the block con-

struction process and don't need to exhaustively calculate all the consistency possibilities. For seed fragments, we choose them from two closest sequences.

3.2 Represent MSA as a Biclustering Problem

Our biclustering procedure has two main steps:

1) Convert to Biclustering Matrix
2) Biclustering with respect to the matrix to identify subgroups of sequences and conserved regions among them.

3.2.1 Convert to Biclustering Matrix

The initial clustering matrix is defined on matrix *M*. The rows of *M* represent the set of block candidates $B=\{B_1, B_2,\ldots,B_n\}$, the columns of *M* represent the set of sequences $S=\{S_1, S_2,\ldots,S_m\}$, and each element M_{ij} of the matrix is set as "1" if the block B_i covers the sequence S_j. Otherwise M_{ij} is set as "0".

3.2.2 Special Case in Biclustering Matrix

Before we apply biclustering to the matrix, we need to first consider a special case: there may exist highly conserved regions (blocks) across all the sequences.

In the biclustering matrix *M*, each row corresponds to a block's coverage on sequences. If a block spans all the sequences, its values in the matrix are all-1s. So before applying biclustering, we check if there are any all-1's rows, note the corresponding blocks as being conserved across all the sequences. We refine the matrix *M* by excluding those all-1's rows.

3.2.3 Bi-clustering With Respect to the Matrix

In this step, the matrix is biclustered. Among many biclustering packages, BiMax (Barkow *et. al.*, 2006) is the most suitable. BiMax doesn't need the users to pre-specify the number of the biclusters. It uses a divide-and-conquer approach to find the inclusion-maximal biclusters. Although it allows overlapped biclusters, it provides a filtering function, which we used to filter results into non-overlapping biclusters.

After applying biclustering, for each obtained bicluster matrix, its columns consist of a subset of sequences, corresponding to a sequence group and its rows consist of a subset of blocks, corresponding to the conserved features of the sequence group. The blocks for each sequence group can be further refined via block assembly and post-processing.

3.2.4 Block Assembly and Post-Processing

After we obtain the blocks for a sequence group, which may be too short, we extend them to both sides until their similarity score falls below a predefined threshold. We also merge two blocks if they are within a relatively short distance. After extending and merging, we can identify an optimal set of non-overlapping blocks. The identification of an optimal set of blocks is actually a classic chaining problem, and can be interpreted as a maximum weight path problem in a directed acyclic graph (Gusfield, 1997). We have implemented a single source DAG shortest path algorithm to find the optimal chain. The time complexity is $O(n^2)$ time for a given set of *n* blocks.

Sometimes a conserved region may not cover all sequences. A block may miss one or two sequences. After we identify all the well-conserved blocks across all the sequences, we can look for those weakly conserved blocks, which may miss a few fragments, and add these blocks back to the optimal chain.

3.3 Recursion

For each sequence group, BlockMSA recursively applies the above two steps 3.1 and 3.2, until each sequence group is small enough or no more conserved regions are found within it.

3.4 Time Complexity

BlockMSA consists of three main procedures. The time for step (1), the identification of block candidates takes $O(nf_sl)$, where n is the number of sequences, f_s is the number of seed fragments and l is the average sequence length in an unaligned region. In step (2), constructing biclustering matrix takes $O(bn)$, where b is number of block candidates. For biclustering, we used BiMax, which takes $O(bnMin(b,n)\beta)$, where β is the number of all inclusion-maximal biclusters. In the BiMax, BlockMSA allows setting up a threshold to limit the number of biclusters, β, to be identified. When the number of sequences is large, we can set a higher threshold to make β smaller. The total time for step (1) and (2) is $O(nf_sl)+O(bnMin(b,n)\beta)$.

Step (3) is a recursion process, which recursively clusters sequences and identifies the blocks in them. This step should be much faster, since we decompose the problems into smaller and smaller problem. This also reflects the advantage of divide-and-conquer, which makes the following step work on smaller problems. The worst running time of BlockMSA is $O(n^2f_sl)+O(bn^2Min(b,n)\beta)$.

4 IMPLEMENTATION

The BlockMSA program is implemented in Java. The pairwise alignments can be obtained from any sequence alignment program. We used ClustalW to do the pairwise alignment. BlockMSA has been tested on Linux, Unix and OS-X. BlockMSA is available under open source licensing and is distributed with documents and examples.

5 RESULTS

ClustalW, MAFFT(L-INS-I), MUSCLE and PROBCONS were run using their default parameter settings. BlockMSA was run using block size of 11 per the following.

5.1 Choice of BlocksMSA Block Size

We evaluated a choice of block size ranging from 3 to 15 on the 322 datasets from BRAliBase and LocaExtR. For each block size, we calculate the mean of SPSs over all the test groups. Block size of 11 gives the best result. (See Fig. 4 and the supplementary data Table 2).

5.2 Evaluation

5.2.1 SPS Scores

The SPS score of each dataset is calculated by using the compalign program (Eddy, 2005). See Fig. 5 and the supplementary

data Table 3. BlockMSA has the leading performance for the test groups for the benchmark tests containing 15 sequences or larger, i.e. *k15* through *k80* of BRAliBase and LocalExtR. BlockMSA demonstrates a trend of increasingly better performance with larger problem size. This trend is contradicted for the T box test set but not the Intron test set. For the three smallest test sets, *k5, k7* and *k10,* MAFFT has the best scores, followed by PROBCONS and then BlockMSA.

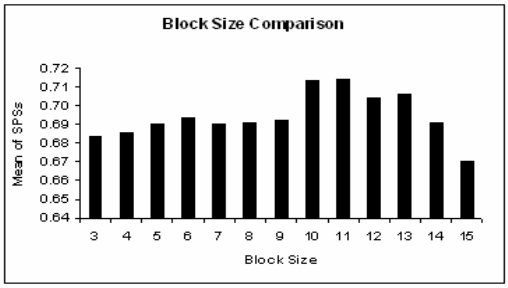

Fig. 4. Block Size Comparison. We test the block size from 3-15 on the test groups from BRAliBase and LocaExtR. For each block size, we calculate the mean of SPSs over all the test groups. Block size of 11 gave the best result.

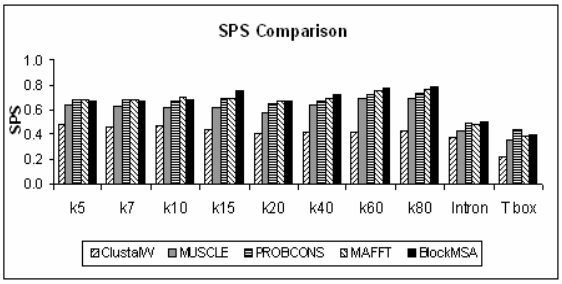

Fig. 5. SPS Comparison. BlockMSA, ClustalW, MAFFT, MUSCLE, and PROBCONS are run on BRAliBase, LocalExtR and LSet. In BRAliBase and LocalExtR, BlockMSA has the leading performance as the input set size increases to 15. For Intron dataset, BlockMSA also leads.

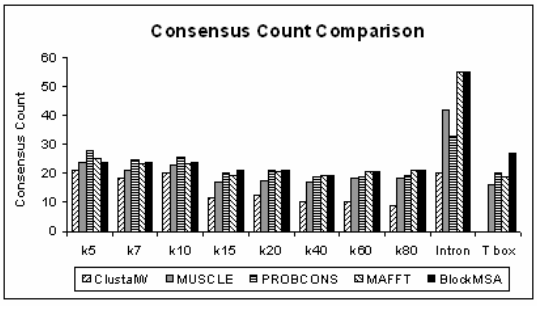

Fig.6. Consensus Count Comparison. BlockMSA, ClustalW, MAFFT, MUSCLE, and PROBCONS are run on BRAliBase, LocalExtR and LSet. For each of the test group, we calculate their average Consensus Count over all the datasets of the test group with the 90% consensus threshold. BlocMSA has leading performance as the number of sequences increases.

5.2.2 Consensus Count

Consensus Counts were calculated using a 90% consensus threshold. See Fig. 6 and supplementary data Table 4. BlockMSA displays the highest Consensus Count for *k15* and ties with MAFFT for the highest consensus count for the benchmark sets larger than 15 sequences. On the smaller problems, PROBCONS has the leading performance. On all the *ki* test groups, except *k5*, all three of these programs attain Consensus Counts no different than 2 from each other, on counts that range from 19 to 25.

BlockMSA also displays the highest Consensus Count for the two biological data sets of LSet. On the T box data, BlockMSA achieves a Consensus Count of 27. Both PROBCONS and MAFFT fall behind, scoring 20 and 19. For the Intron data set, BlockMSA and MAFFT tie for the highest score, 55, followed by MUSCLE, 42, and PROBCONS, 33.

5.3 Biological Validation

BlockMSA demonstrates the best results on the T box data set. BlockMSA identifies the T box motif as a conserved region. It identifies 10 of 12 specifier sequences in the data set. MUSCLE and ClustalW also identify the T box motif, and 9 and 6 specifier sequences respectively. MAFFT and PROBCONS, which score well using SPS and Consensus Count does not identify the T box motif. Their ability to identify the specifier sequences, 5 and 0 respectively, is consistent with their results on the T box motif.

Table 3. T box motif and specifier sequence identification

MSA Program	Identified T box Motif?	Number of Identified Specifier Sequences
BlockMSA	Yes	10
ClustalW	Yes	6
MAFFT	No	5
MUSCLE	Yes	9
PROBCONS	No	0

A T box is defined as identified if a 60% consensus of an alignment produces the motif in its entirety and no gap is inserted within the motif to break its contiguousness. Specifier sequence, (maximum of 12), is defined as identified if all three nucleotides in the codon are present by simple majority consensus.

The entire set of Intron sequences is not yet annotated, thus we can't quantify the results as we did for the T box. Disappointingly, none of these five programs, separately or together, produce a sufficiently palpable alignment for us to promptly annotate the sequences.

We computed consensus sequences and their Consensus Counts for each of the 5 alignments with 4 consensus thresholds (Supplement Table 6). By Consensus Count, MAFFT and BlockMSA continue to be very close in performance for consensus thresholds of 80% or greater. At lower consensus threshold, 70%, MAFFT and MUSCLE hold an edge.

We manually aligned the 20 consensus sequences. From an inspection of that alignment, we are able to make the following qualitative assessment. The 5 alignment programs largely agreed that the group IC1 Introns contain 8 conserved regions. At comparable Consensus Counts, independent of the consensus threshold needed to achieve that count, the 5 programs largely agreed on the location and contents of the conserved region. Thus, qualitatively,

we conclude that BlockMSA, MAFFT, and MUSCLE were better at aligning the Intron sequences than PROBCONS and ClustalW, a conclusion that correlates with Consensus Count.

6. DISCUSSION

This paper is the first effort to represent a MSA problem as a biclustering problem. We incorporate biclustering into a divide-and-conquer approach for local MSA, where conserved blocks of subsequences are identified and further alignment is accomplished by creating sub-problems where the subproblems constitute subsequences of subsets of the sequences. The net result is both a multiple sequence alignment and a hierarchical clustering of the sequences.

The algorithm was tested on ncRNA datasets. ncRNAs pose special problems compared to other sequences. Unlike gene sequences, where single nucleotide polymorphism is often significant, a nucleotide substitution in a helix of a ncRNA is easily compensated by a substitution in the corresponding base pair. Consequently, large homologous sets of ncRNAs often display a high of variability in both length and content.

In our use of biclustering, intuitively, we first perform a large scale search for sets of short, local, multiple alignments, (blocks). Many of these blocks overlap in sequences and suggest different sequence groupings. Rather than simply marking conflicting blocks as mutually exclusive and maximizing the number of consistent blocks, biclustering seeks to maximize the total consistency. Conflicting aspects of the block definitions are simply removed. Blocks, once chosen are not further subdivided. The approach finds and then maintains local areas of agreement. With respect to quantitative measures of MSA, BlockMSA scores comparable to or better than the other leading MSA programs. A contrast surfaces between BlockMSA and the other leading MSA programs in the alignment of large sets of T box and Intron sequences. The other leading MSA programs lag BlockMSA in their ability to identify the most highly conserved regions, and therefore functionally and structurally important parts of the ncRNA sequences. Our conjecture is that PROBCONS and likely MAFFT, which are iterative optimization methods, are introducing gaps to improve SPS score which then results in the break up and misalignment of contiguous conserved regions. It is plausible that this strategy is effective for proteins and coding genes but not ncRNAs. In highly variable ncRNAs, critical (conserved) structures are often flanked by less conserved sequences. These flanks, incorrectly, may provide iterative methods with flexibility to improve score. Without a penalty for breaking up contiguous conserved regions, local optimizations may insert gaps in those contiguous areas. In proteins the larger alphabet size may be enough to limit this effect. In coding genes, misaligning reading frames can also be expected to come at a large price.

Our alignment results depend on the performance of the biclustering program. Currently, biclustering is still an active research area (Cheng *et. al*. 2000; Barkow *et. al.*, 2006). We expect as the biclustering techniques improve and mature, the performance of BlockMSA will improve.

ACKNOWLEDGEMENTS

The authors acknowledge support from the National Science Foundation, DBI-0241180, IIS-0325116 and the National Institutes of Health, GM067317. We thank Tina Henkin for sharing her T box alignment.

REFERENCES

Bailey, T.L. and Elkan, C. (1994) Fitting a Mixture Model by Expectation Maximization to Discover Motifs in Biopolymers. *Proc. Int. Conf. Intell. Syst. Mol. Biol*. 2, 28-36.

Barkow, S., Bleuler, S., Prelic, A., Zimmermann, P., and Zitzler, E. (2006) BicAT: A Biclustering Analysis Toolbox, *Bioinformatics*, 22(10), 1282-1283

Cannone J.J, Subramanian S., Schnare M.N., Collett J.R., D'Souza L.M., Du Y., Feng B., Lin N., Madabusi L.V., Müller K.M., Pande N., Shang Z., Yu N., and Gutell R.R. (2002). The Comparative RNA Web (CRW) Site: An Online Database of Comparative Sequence and Structure Information for Ribosomal, Intron, and Other RNAs. BioMed Central Bioinformatics. 3:2.

Cech,T.R. (1990) Self-splicing of group I introns. *Annu. Rev. Biochem.*, 59, 543–568

Cheng, Y, Church, G.M. (2000). Biclustering of Expression Data. *Proc. of the 8th Int. Conf. on Intelli Syst. for Mol. Biol*., 93–103.

Dhillon, I.S (2001) Co-Clustering Documents and Words Using Bipartite Spectral Graph Partitioning. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 269-274.

Do, C.B., Mahabhashyam, M.S.P., Brudno, M., and Batzoglou, S (2005). PROBCONS: Probabilistic Consistency-based Multiple Sequence Alignment. *Genome Research* 15: 330-340. (http://PROBCONS.stanford.edu/download.html)

Economist (2007). Really New Advances. (http://www.economist.com/science/displaystory.cfm?story_id=9333471)

Eddy S. SQUID – C function library for sequence analysis. (http://selab.wustl.edu/cgi-bin/selab.pl?mode=software#squid).

Edgar, R.C. (2004), MUSCLE: multiple sequence alignment with high accuracy and high throughput, *Nucleic Acids Research* 32(5), 1792-97 (MUSCLE Version 3.6. http://www.drive5.com/MUSCLE/download3.6.html).

Griffiths-Jones, S., Moxon, S., Simon, Marshall, M., Khanna, A., Eddy, S.R., and Bateman A. (2005) Rfam: Annotating Non-coding RNAs in Complete Genomes. *Nucleic Acids Res*. 33, 121-124.

Grundy, F.J. and Henkin, T.M (2003) The T Box and S Box Transcription Termination Control Systems. *Frontiers in Bioscience*, 8, 20-31.

Gusfield, D. (1997) Algorithms on Strings, Trees, and Sequences. *Cambridge University Press*, Cambridge, UK.

Gutell R.R., Lee J.C., and Cannone J.J. (2002). The Accuracy of Ribosomal RNA Comparative Structure Models. Current Opinion in Structural Biology, 12:301-310.

Gutell R.R., Larsen N., and Woese C.R. (1994). Lessons from an Evolving Ribosomal RNA: 16S and 23S rRNA Structure from a Comparative Perspective. Microbiological Reviews 58:10-26.

Katoh K., Kuma K., Toh H. and Miyata T. (2005) MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res.* 33:511-518.

Lawrence. C.E., Altschul, S.F., Boguski. M.S., Liu, J.S., Neuwald, A.N. and Wootton, J. (1993) Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment, *Science*, 262, 208-214

Madeira, S.C. and Oliveira, A.L. (2004) Biclustering Algorithms for Biological Data Analysis: A Survey. *IEEE/ACM Transactions On Computational Biology and Bioinformatics*. 1(1), 24-25

Morgenstern, B. (1999) DIALIGN 2: Improvement of the Segment-To-Segment Approach to Multiple Sequence Alignment. *Bioinformatics* 15(3), 211-218.

Notrdame, C. (2002) Recent Progress in Multiple Sequence Alignment: A Survey. *Pharmacogenomics* 3(1), 131–144.

Prelic , A., Stefan, B. , Philip, Z., Anja, W., Bühlmann, P., Gruissem, W., Hennig, L., Thiele, L. and Zitzler, E.(2006) A Systematic Comparison and Evaluation of Biclustering Methods for Gene Expression Data, *Bioinformatics* 22(9), 1122-1129

Thompson, J.D., Higgins D.G. and Gibson T.J. (1994) CLUSTAL W: Improving the Sensitivity of Progressive Multiple Sequence Alignment through Sequence Weighting Position-Specific Gap Penalties and Weight Matrix Choice. *Nucleic Acids Res*. 22, 4673-4690.

Thompson,,J.D., Plewniak, F. and Poch, O.(1999) BAliBASE: A Benchmark Alignment Database for the Evaluation of Multiple Alignment Programs. *Bioinformatics*, 15, 87–88.

Wilm, A., Mainz, I. & Steger, G. (2006). An enhanced RNA alignment benchmark for sequence alignment programs. *Algorithms Mol. Biol.* 1, 19.