RBT-Km: K-MEANS CLUSTERING FOR MULTIPLE SEQUENCE ALIGNMENT
TECHNICAL REPORT 627

JAVID TAHERI, ALBERT Y. ZOMAYA,
AND BING BING ZHOU
SCHOOL OF INFORMATION TECHNOLOGIES
THE UNIVERSITY OF SYDNEY

JULY 2008
RBT-Km: K-Means Clustering for Multiple Sequence Alignment

Javid Taheri, Albert Y. Zomaya*, Bing Bing Zhou

School of Information Technologies, J12

The University of Sydney, NSW 2006, Australia

{J.Taheri, A.Zomaya, B.Zhou}@usyd.edu.au

* Corresponding author.
Abstract - This paper presents a novel approach for solving the Multiple Sequence Alignment (MSA) problem. K-Means clustering is combined with the Rubber Band Technique (RBT) to introduce an iterative optimization algorithm, known as RBT-Km, to find the optimal alignment for a set of input protein sequences. In this technique, the MSA problem is modeled as a RB, while the solution space is modeled as a plate with several poles corresponding locations in the input sequences that are most likely to be correlated and/or biologically related. K-Means clustering is then used to discriminate biologically related locations from those that may appear by chance. RBT-Km is tested with one of the well-known benchmarks in this field (BALiBASE 2.0). The results demonstrate the superiority of the proposed technique even in the case of formidable sequences.

*Keywords* — K-Means Clustering, Multiple Sequence Alignment, Optimization.
I. INTRODUCTION

Sequence alignment algorithms are techniques that are used to find similarity among several DNA/Protein sequences. These algorithms are classified into two main categories: Pair-wise and MSA algorithms, each designed for special purposes. In Pair-wise algorithms, the main goal is to find similar or closely related parts (motifs) of two sequences; whereas in MSA, the aim is to find the coinciding parts of more than two sequences. Thus, Pair-wise algorithms are mainly used to find similar sequences in a database while MSA methods are employed to find the relationship among several sequences.

Several algorithms and techniques have already been suggested to solve this problem for each of the above two categories. Among these techniques, there exist several classical methods, like Dynamic Programming, that can always find the optimal alignment for any two sequences (Pair-wise). However, these techniques cannot always be generalized to MSA cases (due to the excessive computation that is incurred after the addition of each extra sequence). Therefore, using classical methods in the MSA case is almost impossible. In fact, because it has been shown that MSA is NP-Complete [1]—heuristics are mainly used to solve this problem.

Regardless of the solution methodology, MSA methods can be categorized into three main solution categories: exact, progressive and iterative [2]. In exact methods, which are usually generalized methods of the Needleman and Wunsch algorithm [3], all sequences are aligned to simultaneously find the optimal answer. The main drawback of this class of algorithms is their massive computational requirements which make it impossible to find the answer in polynomial time. In progressive algorithms, sequences are first aligned two-by-two (using an appropriate Pair-wise algorithm) before finding the final alignment. Then, an alignment guidance tree is generated based on the Pair-wise alignment scores. Starting with the closest two, sequences are combined step by step to find the optimal answer. In this case, current sequences are modified to find the best fit for a new combination of sequences. Although this class of algorithms normally manages to find reasonable alignments (especially for generating phylogeny trees), their main disadvantage is their sensitivity for getting trapped into local minima. In iterative methods, all sequences are aligned simultaneously. By using one or more heuristic algorithms, an initial answer is computed first. Then, this initial answer is improved iteratively by using intelligent routines designed for this type of MSAs. Although these algorithms are not as sensitive as progressive algorithm to falling into local minima, however, they have their own drawbacks. For example, the accuracy of the final answer is greatly dependent on the quality of the seed solution.
Based on the above approaches, a number of alignment algorithms are designed to solve the MSA problem, such as MULTALIGN [4], MULTAL [5], PILEUP [6] and CLUSTALX [7], which provides a graphical interface for CLUSTALW [8]. They all use a global alignment algorithm in [3] to construct an alignment for the entire length of the sequences. The main differences between these methods is in the order they combine the input sequences. MULTAL deploys a sequential branching method to align the two closest sequences before building up the final alignment by subsequently aligning the next closest sequence to it. MULTALIGN and PILEUP construct a guide tree using UPGMA [9]. This tree is then used to align larger and larger groups of input sequences. CLUSTALX that uses the alternative neighbor-joining algorithm [10] to construct a guide tree has one of the most sophisticated scoring systems. It considers sequence weighting, position dependant gap penalties, and the automatic switching among scoring matrices based on the degree of similarity among the input sequences. PIMA [11] uses a local DP algorithm to align only the most conserved motifs. Two versions of this method have been developed, ML_PIMA and SB_PIMA, and they differ in the way they order the combining of input sequences and maximum linkage and sequential branching algorithms. DIALIGN [12] employs local alignment based on segment-to-segment comparison to construct the final alignment. Then, an iterative procedure is deployed to combine these segments toward generating the final alignment. PRRP [13] iteratively divides the input sequences into two groups and then subsequently realign them using a global group-to-group alignment algorithm. SAGA [14] evolves a population of alignments in a quasi evolutionary manner to gradually improve their fitness. MAFFT [15] identifies the homologous regions by a Fast Fourier Transform (FFT) approach. Using its simplified scoring matrix, MAFFT manages to significantly reduce the CPU time and increases the accuracy of alignments even for sequences having large insertions and extensions as well as distantly related sequences of similar length. ProbCons [16], which computes posterior-probability matrices and expected accuracies for each Pair-wise comparison, applies the probabilistic consistency transformation, and then computes an expected accuracy guide tree to progressively generate the final alignment. T-Coffee [17] pre-processes a data set of all pair-wise alignments between the input sequences to generate a guide tree for the progressive alignment. T-Coffee not only does focus on the next aligned sequences but also on the whole set of input sequences. MUSCLE [18] as one of the very fast algorithms in this field has three stages: draft progressive, improved progressive, and refinement. At the completion of each stage, a multiple alignment is available and the algorithm can be terminated. The first stage builds a progressive alignment, the second stage that might be iterated attempts to improve the tree and builds a new progressive alignment.
according to this tree, and, the third stage performs iterative refinement using a variant of tree-dependent restricted partitioning. MUMMALS [19] uses probabilistic consistency and improves its alignment quality by using Pair-wise alignment Hidden Markov models (HMMs). Parameters for such models have been estimated from a large library of structure-based alignments. There are also other HMMs methods that use statistical models of the primary structure consensus to align input sequences [20, 21]. HMMT [22] uses the simulated annealing algorithm to maximize the probability that an HMM represents the sequences to be aligned. RBT [23, 24] is another iterative algorithm that uses the \( n \)-dimensional version of the DP table (\( n \) is the number of input sequences) to find the best alignment among input sequences. The analogy of a Rubber Band is a unique contribution of this work.

In this paper, a novel approach, RBT-Km, is presented to solve the MSA problem. The rest of the paper is structured as follows. Section 2 provides the problem statement. Sections 3 and 4 explain main concepts of K-Means clustering and RBT algorithm, respectively. In section 5, details of how Km-Means and RBT are combined to solve the MSA problem are given. Section 6 represents the simulation results. Discussion and analysis followed by conclusion are presented in sections 7 and 8, respectively.

II. MULTIPLE SEQUENCE ALIGNMENT

Let \( \{S_1, S_2, \ldots, S_N\} \) be \( N \) sequences over the alphabet set \( \Psi \), which contains 4 and 20 characters for DNA and Proteins sequences, respectively. Also, let \( \Psi' = \Psi \cup \{-\} \) be the superset of \( \Psi \) with an extra character for a 'gap'. The MSA problem can be defined as finding \( \{S'_1, S'_2, \ldots, S'_N\} \) with the following properties:

1. \( S'_i = S_i \) for all \( i = 1,2,\ldots,N \) providing that all gaps are removed from \( S'_i \).
2. \( |S'_1| = |S'_2| = \cdots = |S'_N| \) where \( |S'_i| \) denotes the length of \( S'_i \).
3. The alignment score, \( \text{Score}(S') = \sum_i \sum_j \text{sim}(S'_i, S'_j) - \sum_i g(S'_i) \) is maximized where \( \text{sim}(S'_i, S'_j) \) denotes a quotation of similarity between \( S'_i \) and \( S'_j \), and, \( g(S'_i) \) is related to gaps of \( S'_i \).

Based on the above, the MSA can be formulated as an optimization problem. However, it is important to note that the complexity of the problem increases exponentially as we add more sequences to the input sequences set—finding the optimal answer is not always possible. Thus, this is why classical methods like DP and Needleman’s
algorithm can only deal with a small number of short sequences.

III. K-MEANS CLUSTERING TECHNIQUE

The K-means clustering algorithm partitions a collection of $n$ vectors $x_j; j = 1, ..., n$ into $c$ groups $G_i; i = 1, ..., c$ and finds a cluster centre in each group so that the dissimilarity cost function is minimized as follows [25, 26]:

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \left( \sum_{k:x_k \in G_i} \| x_k - c_i \|^2 \right)$$

Where $J_i = \sum_{k:x_k \in G_i} \| x_k - c_i \|^2$ is the cost function within group $i$. Here, the partitioned groups are typically defined by a $c \times n$ binary membership matrix $U$, where the element $u_{i,j}$ is ‘1’ if the $j$-th data (i.e., $x_j$) belongs to group $i$, and is ‘0’ otherwise, i.e.,

$$u_{i,j} = \begin{cases} 
1 & \| x_j - c_i \|^2 \leq \| x_j - c_k \|^2 \\
0 & k \neq j \\
& \text{otherwise}
\end{cases}$$

The membership matrix $U$ has the following properties:

1. $\sum_{i=1}^{c} u_{i,j} = 1 \quad \forall j = 1, ..., n$
2. $\sum_{i=1}^{c} \sum_{j=1}^{n} u_{i,j} = n$

The group centers, $c_i$’s, are updated after each clustering step as follows.

$$c_i = \frac{1}{|G_i|} \sum_{k:x_k \in G_i} x_k$$

where

$$|G_i| = \sum_{j=1}^{n} u_{i,j}$$

The following procedure shows the overall view of this clustering technique.
Step 1: Let \( X = (x_1, x_2, \ldots, x_N) \) contains the input data to be clustered, where \( N \) is the number of data points.

Step 2: For all \( x_i \)'s, let each \( x_i \) be a cluster centre.

Step 3: Delete repetitious cluster centers.

Step 4: Repeat Steps 5–7 for all \( x_i \)'s.

Step 5: Let \( m \) be the cluster number that \( x_i \) already belongs to.

Step 6: Find the nearest cluster centre to \( x_i \), and let \( n \) be this cluster number.

Step 7: If \( m \neq n \), remove \( x_i \) from \( m \), add \( x_i \) to \( n \), and update both clusters.

Step 8: Repeat Steps 4–7 until no further \( x_i \) is transferred from one cluster to another.

IV. **Rubber Band Technique**

The Rubber Band Technique is an iterative heuristic to solve the MSA [23, 24]. In this approach, which is inspired by the general behavior of a rubber band on a plate with poles, an initial answer is generated before launching the main optimization procedure. Using several operators, this initial answer is modified iteratively to obtain better alignment scores. The following definitions are essential for the clarification of this optimization procedure.

**A. Grid Answer Space**

The Grid Answer Space (GAS), which is the extended version of the grid table used in DP for Pair-wise alignment, is a multi-dimensional table with a sequence placed in one of its axes. The use of this table provides a unique one-to-one relationship between any possible answer of a MSA and the associated arrowed line as depicted in Fig. 1.

**FIG 1 HERE**

**B. Rubber Band**

Any answer for a MSA can be presented by one and only one arrowed line. This unique arrowed line for each answer is called a Rubber Band (RB).

**C. Primary Pole**

A Primary Pole (PP) refers to a fixed point in GAS that the RB is ought to pass by. In fact, PPs are the sections of
the GAS that force the optimization procedure to align a predefined number of characters (of each sequence) with each other.

D. Secondary Pole

Secondary Poles refer to grid points in GAS that a RB passes through one-by-one to generate the final answer. Now, PPs can be far apart from one other, however, secondary poles need to be adjacent. This type of poles is only used to connect PPs to each other. For brevity purposes, secondary poles are referred to as ‘poles’ for the rest of this paper.

E. Primary Pole Score

As described earlier, each PP points out predefined locations of input sequences that need to be aligned with each other. If the related to these locations are augmented in a single string, the Primary Pole Score for that particulate PP is defined as the alignment score of that augmented string (with respect to the scoring matrix used for the whole alignment).

F. Sticky Poles

Sticky Poles (SPs) are imaginary poles in the system related to locations in a GAS with high Primary Pole Scores. That is, the optimization procedure can have a pole with a high Primary Pole Score if it places a PP on that special place. Therefore, each SP is located in a GAS to represent a column from the input sequences to align identical or closely related nucleotides from different input sequences with one another.

G. Alignment Score

In each MSA instant, an Alignment Score is defined to evaluate the quality of each answer. The Sum-of-Pairs Score (SPS) with Penalized Gap Opening is the criterion used in this approach. In SPS, each column in an alignment is scored by summing the scores of all pairs of characters in that column. The score of the final alignment is then summed over all column scores. In the Penalized Gap Opening scheme, there are two factors to calculate the score/cost of a gap: opening and extension. Gap opening is applied to each gap once and the gap extension corresponds to the length of each gap. The cost of a gap opening is usually considered to be 5–10 times more than that of a gap extension [23, 24]. The use of two factors in calculating a gap is related to a well-known biological fact that having few longer gaps is more plausible than having several short gaps in an alignment.
H. Rubber Band Technique Operators

The main optimization process of RBT consists of various operators that are launched either one or several times to iteratively improve the quality of an initially generated alignment. These operators and their order of deployment are as follows:

Step 1: Initialize Sticky Poles
Step 2: For a predefined number of Tries repeat Steps 3-7
    Step 3: Move Blocks
    Step 4: Move Primary Poles
    Step 5: Jump Primary Poles
    Step 6: Jam Primary Poles
    Step 7: Align Primary Poles
    Step 8: Fine Tuning

V. K-Means in Multiple Sequence Alignment

Finding the locations, in different input sequences, that are most likely biologically related, has a direct impact on the success of a RBT algorithm [23], [24]. For example, in RBT-I, these locations, SPs, are estimated based on similar amino acids indexed in the input proteins. In RBT-L, these SPs are calculated based on the relative locations of similar amino acids in different sequences. However, after close examination of the quality of these SPs, in both versions of RBT, it had been noted that SPs that resemble motifs in an alignment are usually grouped together. Therefore, RBT-Km is designed to identify these groups and discriminate them from singular SPs that might appear by chance. The removal of singular SPs will help the RBT algorithm to have more of such groups in its final alignment. This should yield to finding more motifs and hopefully provide better biologically meaningful alignments. The overall procedure of RBT-Km is as follows:

Step 1: Locate All Sticky Poles
Step 2: Use K-Means Clustering to identify large clusters.
Step 3: Delete small clusters from Sticky Poles set.

Step 4: For a predefined number of Tries repeat Steps 3–7

Step 5: Move Blocks

Step 6: Move Primary Poles

Step 7: Jump Primary Poles

Step 8: Jam Primary Poles

Step 9: Align Primary Poles

Step 10: Fine Tuning

VI. SIMULATION RESULTS

Normally, when solving MSA problems the optimal answer is unknown and there is no concrete criteria to evaluate the quality of a given algorithm, unlike the case for Pair-wise alignment where an optimal solution can always be found. Therefore, standard benchmarks, like BALiBASE, are provided to measure the efficiency of MSA algorithms.

The first version of BALiBASE [27] was dedicated to the evaluation of multiple alignment programs and was divided into five hierarchical reference sets of: (1) equidistant sequences with various levels of conservation, (2) families aligned with a highly divergent ‘orphan’ sequence, (3) subgroups with < 25% residue identity between groups, (4) sequences with N/C-terminal extensions, and (5) internal insertions. For release 2.0 of BALiBASE, these alignments have been manually verified and corrected by superposition of all known three-dimensional structures, using the lsqman program [28]. In this benchmark, an open source program is also provided to score the quality of each answer by comparing it with the one found manually. The maximum score is 1.0 and is assigned to the alignments that are identical to the benchmark’s answer; minimum is 0.0 and is assigned to unrelated/unrealistic answers; and, a number between 0.0 and 1.0 for the others. The score would be higher when the generated answer is closer the manually calculated one.

To demonstrate the performance of the proposed approach, two versions of RBT-Km are used to solve all benchmarks from Reference #2 and #3 of BALiBASE 2.0 [29]. RBT-I-Km and RBT-L-Km are used to classify SPs before running RBT-I and RBT-L, respectively. Therefore, for the rest of this paper, RBT-Km will address both of these versions (unless only one of them is explicitly addressed). For all benchmarks, the BLOSUM62 scoring matrix
with the gap penalty of 10 and 1 selected for the Gap Opening and Gap Extension, respectively. Random walk and homogenous initialization mode is chosen for short/medium (e.g. 1idy and 1tvxA in Reference #2) and long (e.g. 2myr and 1lvl in Reference #2) sequences, respectively. Fig. 2 shows the bar graph representation of the performance of RBT-Km compare with other approaches (including our previous approaches RBT-I [23] and RBT-L [24]) formerly designed to solve the stated benchmarks in Reference #2. Fig. 3 presents similar results for Reference #3. For each benchmark the RBT-Km is executed ten times. For RBT-L and RBT-L-Km, the maximum, average, and minimum of these ten runs are separately depicted to show the robustness of these methods. For RBT-I and RBT-I-Km, the best run of these ten executions is shown; because the results of the other nine runs were not so different from the best one (always within %5 from the final alignment score). Thus, they were omitted from Figs 2 and 3. The reason for obtaining different outcomes for the different runs is significantly related to the nature of the optimization process and its operators. Therefore, it is not surprising they will always find different solutions but these solutions tend to be close enough which is a sign of the algorithm’s robustness and repeatability.

FIG 2 HERE

FIG 3 HERE

VII. DISCUSSION AND ANALYSIS

The results obtained by using RBT-Km were quite different and interesting, covering a vast variety of scenarios. The following sections provide more details.

A. Alignment Score vs. BALiBASE score

The first observation made was the inadequate relationship between Alignment Score, which is purely dependent on the Scoring matrix (BLOSUM62 in this case), and the BALiBASE score, which is purely based on biological facts. However, they seemed to be fairly related. In several cases, gaining higher Alignment Scores yields better BALiBASE scores; although, this cannot be always guaranteed. To investigate this relationship more, we executed the algorithm with different scoring matrices, gap opening and extension values. In almost all cases, the Alignment Score and BALiBASE score showed the same level of uncorrelation. Nevertheless, it seems that in most cases, alignments with higher Alignment Scores have better BALiBASE score as well. Fig. 4 shows a sample of this uncorrelation for 1wit from Reference#3.
B. RBT-Km and Reference #2

Reference #2 of the BALiBASE 2.0 is dedicated to ‘orphan’ sequences. These sequences are significantly different in the number of sequences and the sequences’ length. For this class of sequences, RBT-Km provided varying performance. As shown in Fig. 2, in several cases (1idy, 1csy, 1txvA, 1ubi and 1ajsA), RBT-Km managed to significantly outperform the existing methods. In several other cases, RBT-Km’s performance was fairly comparable to the others, such as, 1uky, 1txvA and 2trx. And finally, there were cases where RBT-Km did not significantly outperform the existing alignment methods, such as, 1cpt, 2myr and 1lvl.

C. RBT-Km and Reference #3

Reference #3 of the BALiBASE 2.0 is dedicated to subgroups of sequences with < 25% residue identity between groups. Performance of RBT-Km in this category was quite different compared to Reference #2. Here, although RBT-Km manages to outperform few of the existing methods, it could not significantly outperform any of them. Overall, the performance of RBT-Km was fairly better than several algorithms.

D. RBT-Km and BALiBASE Score

One of the notable facts about RBT-Km solutions is the non-zero BALiBASE score at all times. By examining Figs. 2 and 3, it can be seen that several of the existing alignment algorithms have significant performance diversity in their results. For example, Fig. 3 shows that PRRP and ML_PIMA achieved BALiBASE scores of 0.0 and 0.905 for 1idy and 1r69, respectively. In other words, in these two cases, the final alignment of these two algorithms exhibits no biological relationship in one case and almost maximum biological relationship in the other case. In contrast, RBT-Km was always able to identify some biological relationship in the sequences to align. In some cases, it almost found the identical answer, like 1idy in Reference #2. In other cases where the biological relationship in the input sequences was subtle and no method could find a reasonable answer, RBT-Km performed reasonably, like 1ajsA in Reference #3.

E. Overall Performance of RBT-Km

The RBT-Km had a reasonable performance in almost all cases. Although there were instances that some of the existing methods found better solutions, with respect to BALiBASE score, in most cases, the quality of RBT-Km’s
alignments were as good as or better than the other methods. In some cases, it could even significantly outperform existing methods, such as 1idy and 1ubi in Reference #2.

Figs. 5 and 6 show the overall performance of RBT-Km compared to other methods. In these figures, the alignment algorithms are sorted according to the average BALiBASE score throughout the whole benchmark. Fig. 5 shows that RBT-Km slightly outperformed RBT-L and significantly outperformed all other methods in Reference #2, almost 20% better than the third best, SAGA. Fig. 6 shows that RBT-Km is ranked 3 based on its average BALiBASE score in Reference #3, although its average BALiBASE score is just 6% below the best approach, PRRP.

\[\text{FIG 5 HERE}\]

\[\text{FIG 6 HERE}\]

\section*{F. RBT-Km versus RBT-I/L}

Figs. 2-5 show that the performance of RBT-I and RBT-I-Km are very similar. Although, the RBT-I managed to obtain better averages in both references, the maximum BALiBASE score for RBT-I-Km is slightly better. However, it seems clustering SPs that are generated based on the AA indices is not very effective in improving an alignment.

The comparison of RBT-L and RBT-L-Km, on the other hand, shows different observations. Figs. 2-5 also show that RBT-L-Max and RBT-L-Avg, which represents the alignment with maximum and average BALiBASE score out of these ten runs, respectively, are fairly equal to RBT-L-Km-Max and RBT-L-Km-Avg, respectively.

However, RBT-L-Km-Min significantly outperforms RBT-L-Min. This implies that having so many SPs, biologically meaningful or not, increases the chance of skewing the optimization process. It also shows that, in several cases, clustering the SPs to identify biologically related motifs, leads to loss of several crucial SPs. These observations imply that the way in which SPs are generated at the start of the main optimization process is the key to better alignments. In conclusion, one can state that the location of an AA is more important than its index in a protein.

\section*{VIII. CONCLUSION}

In this paper, a novel approach (RBT-Km) based on the combination of Rubber Band Technique and K-Means clustering is proposed to solve the Multiple Sequence Alignment problem. In this approach, K-Means clustering is
used to identify locations in the input sequences where there is a higher likelihood of finding are more biologically related (motifs). RBT is then used to find the best possible alignment. RBT-Km is tested by employing several benchmarks from BALiBASE 2.0. The results showed great promise of the proposed approach.

IX. SOFTWARE AVAILABILITY

RBT-Km is written in C++ under Microsoft Visual Net 2005 for Windows Operating System. This algorithm is a part of “ProteinAlignment” software that is developed to implement several multiple alignment algorithms. The web-based version of this software has not been implemented yet, however, researchers can have a free setup package of this software by contacting the authors.

REFERENCES


