MANGO: A NEW APPROACH TO MULTIPLE SEQUENCE ALIGNMENT

Zefeng Zhang and Hao Lin

Computational Biology Research Group, Division of Intelligent Software Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
Email: {zhangzf,linhao}@ict.ac.cn

Ming Li*

David R. Cheriton School of Computer Science, University of Waterloo, Ont. N2L 3G1, Canada
**Email: mli@uwaterloo.ca*

Multiple sequence alignment is a classical and challenging task for biological sequence analysis. The problem is NP-hard. The full dynamic programming takes too much time. The progressive alignment heuristics adopted by most state of the art multiple sequence alignment programs suffer from the 'once a gap, always a gap' phenomenon. Is there a radically new way to do multiple sequence alignment? This paper introduces a novel and orthogonal multiple sequence alignment method, using multiple optimized spaced seeds and new algorithms to handle these seeds efficiently. Our new algorithm processes information of all sequences as a whole, avoiding problems caused by the popular progressive approaches. Because the optimized spaced seeds are provably significantly more sensitive than the consecutive k-mers, the new approach promises to be more accurate and reliable. To validate our new approach, we have implemented MANGO: Multiple Alignment with N Gapped Oligos. Experiments were carried out on large 16S RNA benchmarks showing that MANGO compares favorably, in both accuracy and speed, against state-of-art multiple sequence alignment methods, including ClustalW 1.83, MUSCLE 3.6, MAFFT 5.861, ProbConsRNA 1.11, Dialign 2.2.1, DIALIGN-T 0.2.1, T-Coffee 4.85, POA 2.0 and Kalign 2.0. MANGO is available at http://www.bioinfo.org.cn/mango/.

1. Introduction

Multiple sequence alignment is a basic and essential step of many sequence analysis methods[6]. For example, the multiple sequence alignment is used in phylogenetic inference, RNA structure analysis, homology search, non-coding RNA (ncRNA) detection and motif finding. For recent reviews in this area, see Refs. 35 and 13.

Finding the optimal alignment (under the SP score with affine gap penalty) for multiple sequences has been shown to be NP-hard[44]. A trivial solution by dynamic programming takes $O(n^k)$ time with k sequences, each of length n. Under moderate assumptions[a], the problem has a polynomial time approximation scheme (PTAS)[28]. However, this PTAS remains to be a theoretical solution since it has a high polynomial power related to the error rate. With the rapid growth of molecular sequences, the problem becomes more prominent.

Thus many modern alignment programs resort to heuristics to reduce the computational cost while sacrificing accuracy. The prevailing strategy is the progressive alignment method[14, 41], implemented in the popular ClustalW[41] software, as well as in the more recent multiple sequence alignment programs MUSCLE[11], T-Coffee[34], MAFFT[19], and Progressive-POA[17], to name a few. The idea behind progressive alignment is to build the multiple sequence alignment on the basis of pairwise alignments under the guidance of an evolutionary tree. A distance matrix is computed from similarities of sequence pairs, according to which a phylogenetic tree is built[38]. The multiple alignment is then constructed by aligning two sequences or alignment profiles along the phylogenetic tree. In this way, the progressive alignment method avoids the exponential search.

For a large number of sequences, distance matrix calculation can be slow, and the optimal phylogenetic tree construction itself, under the usual assumptions of parsimony, max likelihood, or max number of quartets, is NP-hard anyways. After all, sometimes the purpose of doing multiple sequence alignment is

*Corresponding author.
[a]For example: when the average number of gaps per sequence is a constant, the problem has a PTAS.

to construct a phylogenetic tree itself. Then if we didn't believe the initial phylogeny (constructed to do multiple sequence alignment), why should we believe in the phylogeny that is constructed based on a multiple alignment which in turn is based on the untrusted phylogeny? Again, heuristics were used to accelerate the phylogenetic tree construction. Pairwise similarity is estimated using fast k-mer counting in MUSCLE, and similar strategy can be seen in the fast version of ClustalW. However, in spite of its most attractive virtue — the speed, progressive approach (adding sequences greedily to form multiple sequence alignment) is born with the well-known pitfall that, error introduced in early stages cannot be corrected later (so-called 'once a gap, always a gap'). Many efforts have been made to remedy this drawback to enhance the accuracy of final alignment. MUSCLE adds tree refinement after progressive alignment stage to tune the result. T-Coffee uses consistency-based score reflecting global information to assist pairwise alignment during the progressive alignment process. PROBCONS adopts a probabilistic consistency-based way. All of them do achieve better accuracy.

There are two alternatives to progressive approaches. One is simultaneous alignment of all sequences by standard dynamic programming (DP). Two packages — MSA[26] and DCA[39] follow this idea. However, algorithms in this category do not scale up because of their heavy computational costs. Another alternative is the iterative strategies[16, 33, 18, 25]. Starting from an initial alignment, these methods iteratively tune the alignment until there are no improvements to the objective functions. These iterative strategies require a good initial alignment as a start point, otherwise the iterative process will be time-consuming or easily fall into local optima.

We now introduce the core idea of MANGO. Given a set of sequences, how do we really judge an alignment? Do we really care about aligning a non-homologous region well? No. What we really care is the aligner's ability of putting similar regions together, including the distant homologous regions. Some similar regions are shared by most of (if not all) sequences, while others may be shared by only a few sequences. Gaps are inserted to get an alignment which lines up the similar regions properly. With this simple observation, we describe a new paradigm to do multiple sequence alignment. Our new algorithm uses the novel idea of optimized spaced seeds, introduced by Ma, Tromp, and Li[31] initially for pairwise alignment, to find similar regions and bind them together via sophisticated algorithms (which are of theoretical interests on their own) and then refine the alignment. Note that similar approaches (Ref. 32) using consecutive or non-optimized k-mers (with some gaps) might have been used in some programs to some degree, however, without optimized spaced seeds, such approaches cannot achieve high sensitivity and specificity as in MANGO. Also note that these optimized spaced seeds are not dependent on any data, they are independently optimized as in Refs. 31, 29 and 9. Our new algorithm requires neither the slow global multiple alignment nor the inaccurate progressive local pairwise alignment.

The optimized spaced seeds have shown their advantages over traditional consecutive seeds for pairwise alignment in PatternHunter software[31]. It has since been adopted by most modern homology search software including BLAST (MegaBLAST). It has been shown[31, 22, 7, 30] that the optimized spaced seeds can achieve much better sensitivity and specificity than consecutive seeds. After that, the idea of a single optimized spaced seed is extended to multiple optimized seeds in PatternHunter II program[29]. Multiple seeds[29, 40], vector seeds[3], and neighbor seeds[8] were studied for even better sensitivity and specificity. One multiple genomic DNA alignment program[4] uses optimized spaced seeds to find hits between two sequences.

To validate our new approach, we have implemented MANGO: Multiple Alignment with N Gapped Oligos. MANGO uses multiple optimized spaced seeds to catch similar regions (hits) quickly with high sensitivity and specificity. A scoring scheme is designed to encode global similarity information in hits. Hits with scores beyond a threshold are arranged carefully to form parts of the alignment. Under the constraint of these hits, banded dynamic programming is carried out to achieve a global solution.

Experiments were carried out on large 16S RNA benchmarks showing that MANGO compares favorably, in both accuracy and speed, against state-of-art multiple sequence alignment methods, including ClustalW, MUSCLE, MAFFT, ProbConsRNA, Dialign, DIALIGN-T, T-Coffee, POA and Kalign.

The experiments were performed only on nu-

cleotide sequences for the purpose of justifying our new approach. For multiple protein sequence alignment, other factors, such as similarity scoring schema and secondary structure information affect the alignment quality a lot, hence potentially would blur the comparative results on the effectiveness of optimized spaced seed approach.

2. Method

The work flow of MANGO is given in Fig 1. Our strategy contains three stages. After any stage, MANGO can be stopped and it will output the alignment constructed so far. In stage one of template construction, MANGO locates super motifs in input sequences, and builds a skeletal alignment by pasting each sequence to the template exposed by motifs. In stage two of hit binding, MANGO first sorts the hits according to their agreements among themselves and then tries to bind hits one by one into the skeletal alignment. Iterative refinement is then carried out in stage three to produce the final alignment, where MANGO picks out one sequence at a time, and aligns it to the current alignment of the rest of sequences.

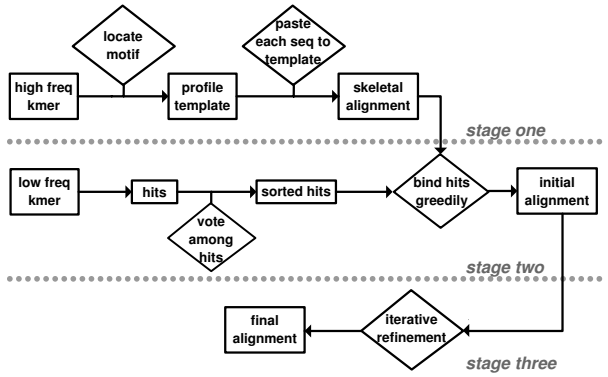

Fig. 1. Three stages of MANGO: template construction, hit binding and iterative refinement, producing skeletal alignment, initial alignment and final alignment, respectively.

2.1. Seeds selection

Following the original notation of Ref. 31, we denote a spaced seed by a binary string. A "1" in the spaced seed means it requires a match at the position, and a "0" indicates a "don't care" position not requiring a match. The length of the seed is the length of the binary string, and the weight of the seed is the number of 1's in the seed. For reasons why the optimized spaced seeds are much better than the consecutive BLAST type of seeds, please see Refs. 31 or 30 which points to many more recent theoretical studies. We have used eight highly independent spaced seeds[b] generated from the parent seed 1110110010110101111, with seed weight 13 and seed length ranged from 19 to 23 from Ref. 9. 82 single optimal spaced seed with weight ranged from 9 to 16 and length ranged from 9 to 22, are optimized against a 64-length IID region of similarity level 0.7, using the dynamic programming algorithm described in Ref. 22. These 82 seeds are sorted with decreasing weight, to prefer specificity to sensitivity. We used the first seed to locate the super motifs and construct profile template, then we applied the other seeds one by one.

For each seed-match (namely a *hit*), we call the matched fragment a k-mer. The k-mer has the same length as the seed.

2.2. Stage one: constructing profile template

If a piece of sequence segment is shared by a considerably large portion of input sequences, we call it a super motif. The super motifs reflect the conserved parts among sequences and are very likely to appear in the final alignment, hence pining them down will give guidance to the whole alignment process. MANGO uses an optimized spaced seed to detect super motifs, thus they are not necessarily the same (due to 'no care' positions of spaced seed), as long as they have high similarity to be caught by the spaced seed. The detection is performed as the following. Highly frequent k-mers (currently defined as 25% of the input sequences having the k-mer) extracted by the spaced seed are lined up according to their relative appearance in each sequence. Then MANGO determines overlapping portion (see middle of Fig 2 for demonstration of overlap) of adjacent k-mers, by searching for their existing overlapping parts in each sequence. In this way, a super motif is represented

[b] 1110011011001010111 1,1101110110000110100111,
1011110010110111011,1100111000001011010111 1,
1011011101011000111 1,1010101011001010010111 1,
1110110001111101101,1100111011001001000111 1.

by a series of overlapping k-mers, and all those k-mers are concatenated together to form the profile template. After that MANGO aligns each sequence to the template. As Fig 2 indicates, those highly frequent k-mers resided inside each sequence is directed (aligned) to corresponding position in profile template, thus produces a skeletal alignment.

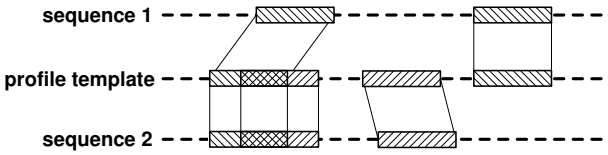

Fig. 2. MANGO locates super motifs by highly frequent k-mers (shaded boxes) and constructs profile template by concatenating them. Those k-mers inside each sequence are aligned to the profile template, producing a skeletal alignment.

The impact of this stage is twofold: the profile template provides anchors and constraints for later alignment process, and wiping out highly frequent k-mers greatly reduces the number of hits to be considered in stage two of hit binding.

2.3. Stage two: binding the hits

2.3.1. *Vote among hits*

After getting rid of the super motifs, less frequent k-mers extracted by single or multiple spaced seeds will generate a set of hits among sequences. Hits may conflict to each other and MANGO tries to select a good compatible subset of them. Since the consistent relationship among those hits reflects the global similarity of input sequences, MANGO encodes global information into each hit by assigning it a priority score, which is voted by other hits to agree or disagree that this hit should appear in final alignment.

The consistency and inconsistency relationships of the hits are illustrated in Fig 3, corresponding to "yes" vote (positive score) and "no" vote (negative score), respectively. Assume that hit_i and hit_j occur between the same two sequences. Let $S(i,j)$ be the vote score for hit_i by hit_j, which is calculated as:

(1) if hit_i and hit_j are incompatible (either they are order incompatible as in Fig 3(1.a) or they are overlapped but their nucleotide mapping order is inconsistent as in Fig 3(1.b)), they can not appear in the same alignment simultaneously. Hence hit_j will vote against hit_i, and $S(i,j) = -W_{\text{disagree}}$;

(2) if hit_i and hit_j are order compatible as Fig 3(2.a) indicates, the appearance of hit_j in a certain alignment will encourage hit_i to appear too, and $S(i,j) = W_{\text{agree}}$;

(3) if hit_i and hit_j are overlapped and their nucleotide mapping order is consistent as Fig 3(2.b) indicates, MANGO further considers their overlapped region size in this case. Define overlapping ratio between them as $\alpha = overlap\_size/(2l - overlap\_size)$, where l is the spaced seed (hit) length. Then $S(i,j) = \alpha * W_{\text{overlap\_high}} + (1-\alpha) * W_{\text{overlap\_low}}$;

We also have indirect votes from k-mers on other sequences too. If hit_j votes for or against hit_i (as in Fig 3(3.a) and (3.b)), those k-mers same as k-mer of hit_j on other sequences (C in Fig 3(3)) will increase the power of voting, since occurrence of C through hit_j will enhance the probability that hit_i appears or doesn't appear in final correct alignment.

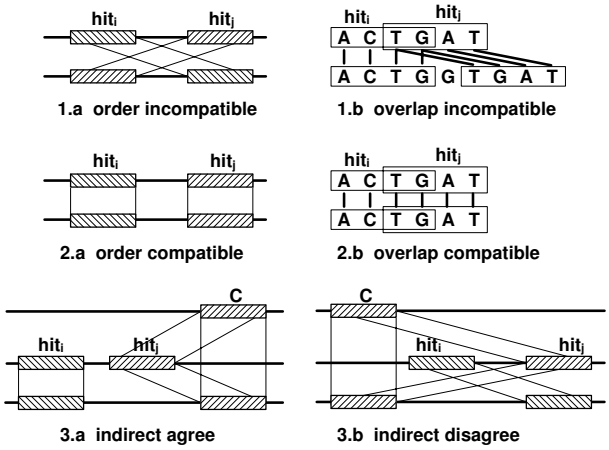

Fig. 3. To uncover global similarities, MANGO assigns hit_i a priority score, which is voted by other hits: (1) "no" vote: hit_j and hit_i are order incompatible in 1.a or they have inconsistent nucleotide mapping order in 1.b; (2) "yes" vote: hit_j and hit_i are order compatible in 2.a or they have consistent nucleotide mapping order in 2.b; (3) indirect vote from k-mer C who increases voting power of hit_j to hit_i.

Let $N(hit)$ be the number of sequences that have the same k-mer as that inside hit. The priority score assigned to hit_i is calculated as $\sum_j S(i,j) * (1 + (N(j) - 2) * W_{\text{indirect}})$. The voting results are collected and hits are sorted according to their score. Low scored hits (probably random hits) are removed

and the remaining hits are considered as candidates into next hit binding stage.

2.3.2. *Bind hits greedily*

In the second part of stage two of hit binding, MANGO will try to bind each hit (high score first) into skeletal alignment generated from stage one, by greedily checking hit candidates one by one. To bind a hit is to align all corresponding sequence letter pairs along the hit and fix their positions (once aligned, they remain aligned). What we want is to arrange the relative positions of the hits carrying similarity information. Thus, a natural way is to formulate an alignment solution as a directed acyclic graph(DAG), by viewing aligned nucleotides as one vertex. We found Ref. 25 employs a similar DAG formation.

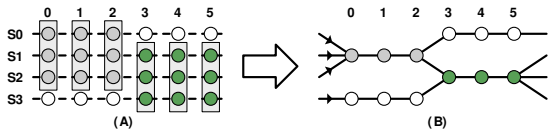

Fig. 4. By viewing aligned nucleotides as one vertex, an alignment can be formulated as a DAG.

Given N sequences each with L_i nucleotides, we denote j^{th} nucleotide in sequence S_i as a vertex $S_{i,j}$. Directed edges are linked from $S_{i,j}$ to $S_{i,j+1}$, for $0 \leq j < L_i - 1$. Thus, the initial graph $G = \langle V, E\rangle$ has $\sum_{0\leq i<N} L_i$ vertices and $\sum_{0\leq i<N}(L_i - 1)$ edges. If the nucleotide of $S_{i,j}$ is aligned to another nucleotide of $S_{i',j'}$, we merge vertex $S_{i,j}$ to $S_{i',j'}$, with their corresponding edges naturally to the super node. It is obvious that each valid alignment solution is equivalent to a DAG. The skeletal alignment we have obtained in the first stage of MANGO is such a DAG, as Fig 4 indicates.

For the DAG $G = \langle V, E\rangle$ of skeletal alignment, a hit can be represented by a vertex pair array $\langle(s_1, t_1), (s_2, t_2), \ldots, (s_l, t_l)\rangle$, where $s_i, t_i \in G(1 \leq i \leq l)$ and $\langle s_i, s_{i+1}\rangle \in E, \langle t_i, t_{i+1}\rangle \in E(1 \leq i \leq l-1)$. In other words, a hit is a vector of adjacent vertex pairs. Binding a hit means to merge a vertex pair into one single vertex. But the success of doing so requires that the resulting graph is still a DAG, which represents a valid alignment solution. Starting from the DAG of skeletal alignment, MANGO greedily chooses a hit from the candidate set. If the above criteria are satisfied, MANGO binds it to the DAG (updates the DAG by merging the corresponding vertices); otherwise, the hit is discarded.

Let $reach(s, t)$ be a predicate which is true iff $s \neq t$ and there is a directed path from s to t in the DAG. If we merge two vertices x and y when $reach(x, y) \vee reach(y, x)$ is true, then the resulting graph is no longer a DAG, due to the circle introduced. So a hit $\langle(s_1, t_1), (s_2, t_2), \ldots, (s_l, t_l)\rangle$ can be bound into alignment DAG, if and only if $\neg reach(s_i, t_i) \wedge \neg reach(t_i, s_i)$, for $1 \leq i \leq l$.

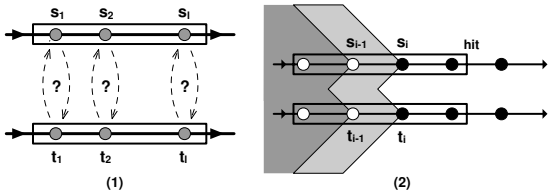

Fig. 5. (1) The success of binding a hit $\langle(s_1, t_1), (s_2, t_2), \ldots, (s_l, t_l)\rangle$ requires that there is no directed path between any vertex pair $(s_i, t_i), 1 \leq i \leq l$; (2) We check that by expanding left predecessors (grey area) of (s_i, t_i), bypassing those already expanded by (s_{i-1}, t_{i-1}) (dark grey area).

Now we describe the process of binding a hit candidate. As indicated in Fig 5(1), we need to check that there is no directed path between two vertices in any vertex pair, and if success, we will update the DAG. An intuitive way is to maintain a transitive closure matrix, that is, to record the reachability for each pair of vertices. This strategy has constant query time and update time of $O(l * n)$, where n is the vertex number. Neither the space complexity of $O(n^2)$ nor the updating complexity of $O(l * n)$ is acceptable, as n is too large and the update operation is quite frequent.

To our knowledge, the best result of reachability algorithm without explicitly maintaining transitive closure matrix on general directed graph is due to Ref. 37. They provide a nice full dynamic (supporting query, edge insertions and deletions) algorithm with query time $O(n)$ and amortized update time of $O(m + nlog(n))$, where m is the edge number. The best result on DAGs appears in Ref. 36 with query time of $O(\frac{n}{log(n)})$ and amortized update time of $O(m)$. Neither of them is suitable to our problem with frequent updates of vertex merging. Thus we design an algorithm supporting query and vertex merging on DAGs, with nearly constant update

(merging) time and $O(n)$ query time in the worst case, for a vector of arbitrary number of adjacent vertex pairs.

Let $L(s) = \{i|reach(i,s), i \in V\}$ and $R(s) = \{i|reach(s,i), i \in V\}$. We call $L(s)$ the (left) predecessor set of s and $R(s)$ the (right) successor set of s. We know that:

Lemma 2.1. *$reach(s_i, t_i) \iff R(s_i) \cap L(t_i) \neq \emptyset$.*

Corollary 2.1. *hit $\langle (s_1,t_1),(s_2,t_2),\ldots,(s_l,t_l)\rangle$ can be bound into alignment DAG $\iff$ $L(s_i) \cap R(t_i) = \emptyset \wedge R(s_i) \cap L(t_i) = \emptyset$, for $1 \leq i \leq l$.*

The reachability of vertex pairs in a hit are not independent, due to:

Lemma 2.2. *$L(s_i) \subset L(s_{i+1})$ and $R(t_i) \supset R(t_{i+1})(1 \leq i < l)$.*

By Corollary 2.1 and Lemma 2.2, we can check reachability of vertex pairs in an incremental way. After we checked that $L(s_i) \cap R(t_i) = \emptyset \wedge R(s_i) \cap L(t_i) = \emptyset$, for vertex pair (s_{i+1}, t_{i+1}), we only need to check that $(L(s_{i+1}) \setminus L(s_i)) \cap R(t_{i+1}) = \emptyset \wedge R(s_{i+1}) \cap (L(t_{i+1}) \setminus L(t_i)) = \emptyset$. So for each given vertex pair, we check reachability between them by expanding the left predecessors of these two vertices, searching for any vertex that is located at the right side (marked black in Fig 5(2)) along two sequences, while bypassing those vertices already expanded by prior pairs. This can be done in $O(n)$ time in the worst case for arbitrary number of vertex pairs, because no vertex is expanded twice during checking process of all vertex pairs of a hit. Nearly constant update (merging) time $O(l * \alpha(n))$, where $\alpha(n)$ is inverse Ackermann function, is achieved if for each vertex we maintain a list of its incoming vertices and record the list head and tail pointers. To merge two vertices, we alias them by Union-Find algorithm and merge their incoming lists.

At this point, MANGO applies seeds sequentially, with seeds of higher weights used first. That is, for each given seed, MANGO searches for hits, calculates their voting score and finds the best way to bind them. This strategy is temporarily adopted mainly for memory efficiency. In future, MANGO will include an option of using several seeds each step. This will increase significantly the memory usage, however it has a potential to further increase sensitivity and specificity.

It should be noted that similar regions are arranged carefully to form a basic alignment without considering any gap penalty in the above two steps.

2.4. Stage three: iterative refinement

After stage two of hit binding, input sequence letters are tightly bound as a DAG, and the final stage of MANGO tries to refine this alignment, with respect to the topological structure of DAG. Each time one sequence is selected and corresponding DAG vertices (black nodes in Fig 6) are picked out to form one input. The rest of vertices inside DAG form another input, and MANGO performs a dynamic programming (DP) using similar heuristic strategy to Ref. 20. User can also choose to perform an optimal alignment search with affine gap penalty described in Ref. 21, which we modified to fit in our alignment situation.

The scoring scheme we used is the simplest sum-of-pairs (SP) score, with matching nucleotide pairs scoring 1 and mismatching pairs scoring 0. The gap open penalty and gap extend penalty are -1.53 and -0.23 respectively, same with that used in MAFFT.

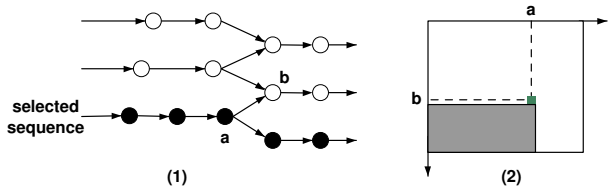

Fig. 6. (1) In refinement stage, MANGO picks out a sequence each time and aligns it to other sequences based on current alignment. (2) The topological order of DAG (such as vertex a must appear before vertex b) helps to narrow down the search space (the grey area is skipped).

However, the vertex orders in DAG can help to narrow down the search space of DP. This can be seen from Fig 6. The DAG requires vertex b to appear after vertex a in any valid alignment, thus a portion (shaded area in Fig 6(2)) of quadratic search space is cut off. In practice, the DP is performed in a banded search space with almost linear time and space.

We find the idea of Iterative POA[25] is similar to what we do in the refinement part. It extracts one sequence out and aligns it to the rest DAG by DP, slightly different from the DP with constraints we revised from Ref. 21. However, without the guidance

of bounded hits, local optima is easily reached in the iterative way.

The above three stages constitute the core algorithms of MANGO. Finding profile template and constructing a skeletal alignment is the fastest; binding hits is efficient by the reachability detection algorithm described above, and experiments (see below) show that this stage is responsible for most accuracy ratio; refinement stage is the slowest, but it can uncover similarities that have escaped the detection of the optimal spaced seeds.

3. Results

This section assesses the performance of MANGO on three large alignment benchmarks of small subunit (SSU) ribosomal RNA (16S RNA) for phylogenetically representative sets of prokaryotes, mitochondrial and eukaryotes, respectively (version 9.36)[5]. These alignments are hand-curated by the Ribosomal Database Project (RDP-II) at the Michigan State University.

All leading multiple sequence alignment software (using their most recent publicized versions), which are able to do large scale multiple sequence alignment, are compared with MANGO: ClustalW 1.83[41], the most popular software; MUSCLE 3.6[11], a fast program aims at speed and accuracy; MAFFT 5.861[19], with many recent improvements from previous versions; Dialign 2.2.1[32]; DIALIGN-T 0.2.1[2], improved version of Dialign; T-Coffee 4.85[34], well known for its accuracy; POA 2.0[25, 17], based on partial order graph formulation and Kalign 2.0[24]. Two versions of ClustalW: ClustalW-fast (with $-quicktree$ parameter) and ClustalW (full version), two versions of MUSCLE: MUSCLE-fast (with $-maxiters$ 1 $-diags$ parameters) and MUSCLE (full version), two versions of MAFFT: MAFFT-fast (FFT-NS-1 with $-retree$ 1 $-maxiterate$ 0 parameters, the fastest version) and MAFFT (L-INS-i with $-localpair$ $-maxiterate$ 1000 parameters, the most accurate version), two versions of Dialign: Dialign-fast (with $-o$ $-ds$ parameters to speed up DNA alignment) and Dialign (full version) are tested to compare speed, memory usage and alignment accuracy. All experiments were carried out on a PC with Intel Celeron-2.0 processor and 1G main memory, and all data sets and experimental results are available on MANGO website. ProbConsRNA[10] has comparable SP score and PS score with MAFFT on the three data sets. However, its consuming time is too long, so we included the MAFFT result only.

3.1. Measurement

Alignment accuracy is measured by SP and PS scores[11]. Let A be the alignment generated by the program. Let R be the (supposedly correct, or human curated in our case) reference alignment. SP score (also known as Q score and SPS score[43]) is defined as the number of correctly aligned nucleotide pairs in A divided by total number of nucleotide letter pairs in R. PS score[12] is defined as the number of correctly aligned nucleotide pairs in A divided by total number of nucleotide pairs in A. Thus, SP score measures the sensitivity of the alignment A and PS score measures the specificity of the alignment A. All these scores were calculated by the program *bali_score* from Ref. 43, which removes noisy columns containing mostly gaps from alignments.

3.2. MANGO versions

Time and accuracy varies if we choose to use first eight seeds or use all ninety seeds. Also time is reduced if we remove refinement stage. We simply provide the experimental results for all four versions of MANGO: MANGO8, where we used eight neighbor seeds without the refinement stage; MANGO8-r, where the refinement stage was added to MANGO8; MANGO90, where we used all 90 seeds without refinement; and MANGO90-r, where the refinement stage was added to MANGO90.

3.3. The 16S SSU rRNA data set

The 16S SSU rRNA benchmark has three data sets for prokaryotes, mitochondrial and eukaryotes, respectively. The first data set (prokaryotes) has 218 sequences with average sequence length 1487 bp; the second data set (mitochondrial) has 76 sequences with average length 1075 bp and the third data set (eukaryotes) has 140 sequences with average length 1823 bp. The mitochondrial data set is the smallest but with the lowest similarity among three data sets.

3.4. The assessment

Table 1 to 3 present the experimental results on the data sets. We make a few observations below:

- On all three data sets, MANGO90 is simultaneously more sensitive (SP score), more specific (PS score) and much faster than ClustalW, ClustalW-fast, MUSCLE, iterative POA and progressive POA. The sensitivity of MUSCLE-fast is too low. MANGO8 has higher sensitivity, higher specificity, and higher speed simultaneously than MUSCLE-fast. Comparing to MUSCLE and ClustalW (the two most trusted software), MANGO90-r achieves both higher SP and PS scores, in half of MUSCLE's or ClustalW's time, on all three data sets.
- Except for a few unbalanced cases (when DIALIGN-T has very low sensitivity but higher specificity), for all data sets, with lower specificity and significantly lower sensitivity, Dialign, DIALIGN-T and Dialign-fast all run many times slower than any version of MANGO. The specificity of these programs is generally good. Although Kalign runs quite fast, it has both low sensitivity and specificity.
- On all three data sets, MANGO has slightly lower sensitivity (SP scores) but higher specificity (PS scores) against MAFFT full version but MANGO (MANGO90-r), at similar sensitivity and specificity, is at least 5 times faster than MAFFT full version in all cases.
- On the low similarity data of mitochondrial data set, all programs have very poor performance except MANGO and MAFFT. MAFFT has high SP score and MANGO has high PS score and with second highest SP score. Here MANGO90 finished in 37 seconds vs. MAFFT finishing in 14 minutes.
- T-Coffee failed in two cases. For the case (mitochondrial) it finally finished after 37 hours, its sensitivity and specificity were both inferior to MANGO90-r, which finished in less than 3 minutes.
- MANGO seeds were trained universally as in PatternHunter[31, 29, 9], not tuned for these data sets.
- MANGO's strategy is more suitable for medium scale input, such as data sets with more than 20 sequences. Because MANGO does not invest heavily on the refinement stage, it doesn't perform as well as some of the specialized methods such as ClustalW on the popular nucleotide MSA benchmark BRALiBASE[15], in which each reference alignment is generally short and composed of small number of sequences.
- MANGO is also able to handle extremely large data sets. In addition to these three data sets, we have also used MANGO to align over 20,000 reads for some repeat regions, for the purpose of sequence assembly, which was one of our original reasons to begin this research.
- As a word of caution, higher SP scores or higher PS scores do not necessarily imply that the alignment is better biologically. What we hope to achieve via this experiment is to demonstrate that our new approach can help to capture the global features (reflected by the PS score) of an alignment. We think capturing sufficient aligned parts with higher confidence is probably more important than aligning more pairs in a random region.

We also carried out experiments on multiple alignment of Alu sequences in Human genome. Due to the space limit, more experimental results will be presented in the full version of this paper.

4. Discussion

In this paper, we have presented a new approach to multiple sequence alignment, orthogonal to existing approaches. Necessary algorithms were developed for managing the hits efficiently, getting rid

Table 1. performance evaluation on prokaryotes 16S SSU rRNA benchmark data set

data set	program	SP	PS	time(s)	mem(M)
prokaryotes	ClustalW-fast	0.937	0.929	324	4
	ClustalW	0.913	0.932	6380	4
	MUSCLE-fast	0.925	0.928	145	216
	MUSCLE	0.937	0.928	1427	216
	MAFFT-fast	0.931	0.930	42	137
	MAFFT	**0.949**	0.942	6006	149
	Dialign-fast	0.924	0.934	123083	409
	Dialign	0.921	0.928	202903	401
	DIALIGN-T	0.770	**0.952**	20719	201
	T-Coffee	-	-	failed	-
	POA-iter	0.881	0.901	1106	48
	POA-prog	0.899	0.927	8495	71
	Kalign	0.926	0.914	73	6
	MANGO8	0.929	0.942	121	49
	MANGO8-r	**0.944**	0.939	660	49
	MANGO90	0.943	**0.945**	299	49
	MANGO90-r	**0.944**	**0.943**	592	49

[a] The SP and PS scores for each method on three 16S SSU rRNA benchmark data sets are listed, together with CPU time in seconds and memory usage in megabytes. Top three SP and PS scores are marked bold.

Table 2. performance evaluation on mitochondrial 16S SSU rRNA benchmark data set

data set	program	SP	PS	time(s)	mem(M)
	ClustalW-fast	0.442	0.408	63	3
	ClustalW	0.478	0.469	242	3
	MUSCLE-fast	0.357	0.364	30	74
	MUSCLE	0.442	0.421	333	74
	MAFFT-fast	0.591	0.558	33	147
	MAFFT	**0.734**	0.691	829	129
	Dialign-fast	0.538	0.777	1070	34
mitoch-	Dialign	0.588	0.763	3024	37
ondrial	DIALIGN-T	0.209	0.000	1202	26
	T-Coffee	0.594	0.740	135063	1000
	POA-iter	0.316	0.328	234	30
	POA-prog	0.485	0.494	581	10
	Kalign	0.375	0.330	12	4
	MANGO8	0.530	**0.859**	4	10
	MANGO8-r	**0.595**	0.739	589	89
	MANGO90	0.590	**0.878**	37	10
	MANGO90-r	**0.596**	**0.836**	155	42

[b] The inconsistency of SP and PS score for DIALIGN-T is due to the calculation process of *bali_score*, which removes columns containing mostly gaps.

Table 3. performance evaluation on eukaryotes 16S SSU rRNA benchmark data set

data set	program	SP	PS	time(s)	mem(M)
	ClustalW-fast	0.874	0.844	348	5
	ClustalW	0.844	0.866	4013	5
	MUSCLE-fast	0.588	0.675	148	186
	MUSCLE	0.863	0.836	2638	186
	MAFFT-fast	0.887	0.874	81	192
	MAFFT	**0.905**	0.880	3428	147
	Dialign-fast	0.821	0.891	28604	205
eukar-	Dialign	0.840	0.894	53791	205
yotes	DIALIGN-T	0.761	**0.936**	15527	112
	T-Coffee	-	-	failed	-
	POA-iter	0.753	0.796	1011	69
	POA-prog	0.796	0.822	5238	76
	Kalign	0.869	0.837	71	6
	MANGO8	0.861	**0.904**	75	28
	MANGO8-r	**0.896**	0.880	2281	72
	MANGO90	0.887	**0.911**	305	29
	MANGO90-r	**0.890**	0.896	661	41

of false hits and combining the hits. To demonstrate our methodology, we have implemented MANGO for multiple alignment of nucleotide sequences.

Our new approach build the alignment "region by region", unlike progressive alignment methods which build the alignment sequence by sequence. In this way, we align sequences simultaneously, avoiding the inherent 'once a gap, always a gap' phenomenon for progressive approaches. Additionally, the new paradigm has two more advantages. First, prior biological knowledge can be easily added to the skeletal alignment, as user-defined constraints/bindings. Secondly, input sequences from diverging families can cause problems for many global multiple sequence alignment programs. In this case, local multiple sequence alignment programs[1, 27] are preferable. This problem is taken care of naturally by our method. Since we build the skeletal alignment and arrange hits step by step on the basis of similar regions, sequences with different conserved regions do not interfere among themselves as much as in the progressive alignment approaches.

Hits produced by spaced seeds are the footstone of our method and the alignment accuracy is based on the correctness of binding hits. Though our voting strategy can get rid of most of the false hits, choosing a better set of multiple spaced seeds with higher sensitivity and specificity can enhance the accuracy and reduce running time. Up to now, the refinement stage contributes not too much to the final result, but we believe changing the scoring scheme in the final stage will enhance the result more.

Though we have focused on nucleotide sequences in this paper for the purpose of clean methodology comparative study, our method can be extended to protein sequences by designing multiple seeds for protein sequences and redefining seed hits on protein sequences similar to protein pairwise alignment[23], then fusing the secondary structure and profile information into hit finding. This project is underway.

Acknowledgement

We thank Bin Ma for providing the neighbor seeds from Ref. 9, and Dan Brown, Dongbo Bu and Yu Lin for discussions on multiple sequence alignment.

References

1. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J. (1990) Basic local alignment search tool. *Journal of Molecular Biology* **215**, 403-410.
2. Amarendran, R.S., Jan W.M., Michael K., Burkhard M. (2005) DIALIGN-T: An improved algorithm for segment-based multiple sequence alignment. *BMC Bioinformatics* **6**, 66.
3. Brejová B., Brown, D., Vinar, T. (2005) Vector seeds: An extension to spaced seeds. *Journal of Computer and System Sciences*. bf 70 364-380.
4. Brown, D.G., Hudek, A.K., (2004) New Algo-

rithms for Multiple DNA Sequence Alignment. *Algorithms in Bioinformatics, 4th International Workshop (WABI)*, 314-325
5. Cannone, J.J., Subramanian, S., Schnare, M.N., Collet, J.R., D'Souza, L.M., Du, Y., Feng, B., Lin, N., Madabusi, L.V., Muller, 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.
6. Carrillo, H. & Lipman, D. J. (1988) The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.* **48**, 1073-1082.
7. Choi, KP. & Zhang, LX. (2004) Sensitivity analysis and efficient method for identifying optimal spaced seeds. *Journal of Computer and System Sciences*, **68**, 22-40.
8. Csűrös, M. & Ma, B. (2005) Rapid Homology Search with Two-Stage Extension and Daughter Seeds. *COCOON* **11**, 104-114.
9. Csűrös, M. & Ma, B. (2006) Rapid homology search with neighbor seeds. *To appear in Algorithmica*
10. 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.
11. Edgar RC. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res* **32**, 1792-1797.
12. Edgar RC. & Sjolander K. (2004) MUSCLE: A comparison of scoring functions for protein sequence profile alignment. *Bioinformatics* **8**, 1301-1308.
13. Edgar RC. & Batzoglou S. (2006) Multiple sequence alignment. *Current Opinion in Structural Biology* **16**, 368-373.
14. Feng, DF. & Doolittle, RF. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.* **25**, 351-360.
15. Gardner PP, Wilm A, Washietl S. (2005) A benchmark of multiple sequence alignment programsupon structural RNAs. *Nucleic Acids Res.* 2005, **33**, 2433-2439.
16. Gotoh O. (1996) Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *J Mol Biol* 1996, **264**, 823-838.
17. Grasso, C. and Lee, C. (2004) Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems. *Bioinformatics* 2004, **20**, 1546-1556.
18. Katoh K, Misawa K, Kuma K, Miyata T. (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res* 2002, **30**, 3059-3066.
19. Katoh K, Kuma K, Toh H, Miyata T. (2005) MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res* 2005, **33**, 511-518.
20. Kececioglu, J. and W. Zhang. (1998) Aligning alignments. *Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching* 189-208.
21. Kececioglu, J. and D. Starrett. (2004) Aligning alignments exactly. *Proceedings of the 8th ACM Conference on Research in Computational Molecular Biology (RECOMB)* 85-96.
22. Keich, U., Li, M., Ma, B., Tromp, J. (2004) On spaced seeds for similarity search. *Discrete Applied Mathematics* **138** 253-263.
23. Kisman, D., Li, M., Ma, B., and Wang, L. (2005) tPatternHunter: gapped, fast and sensitive translated homology search. *Bioinformatics*, 21:4, 542-544.
24. Lassmann T. and Erik L.L. Sonnhammer (2005) Kalign - an accurate and fast multiple sequence alignment algorithm. *BMC Bioinformatics* **6**, 298.
25. Lee C, Grasso C, and Sharlow MF. (2002) Multiple sequence alignment using partial order graphs. *Bioinformatics* 18(3), 452-464
26. Lipman, D. J., Altschul, S. F. & Kececioglu, J. D. (1989) A tool for multiple sequence alignment. *Proc. Natl Acad. Sci.* USA, **86**, 4412-4415.
27. Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F. & Wootton, J. C. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, **62**, 208-214.
28. Li, M., Ma, B., Wang, L. (2000) Near optimal alignment within a band in polynomial time. *Proc. 32nd ACM Symp. Theory of Computing* (STOC'00), Portland, Oregon, 425-434.
29. Li, M., Ma, B., Kisman, D., Tromp, J. (2004) PatternHunter II: highly sensitive and fast homology search. *Journal of Bioinformatics and Computational Biology* **2** 411-439.
30. Li, M., Ma, B., Zhang, L. (2006) Superiority and complexity of the spaced seeds. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithms (SODA 2006)*, 444-453.
31. Ma, B., Tromp, J., Li, M. (2002) PatternHunter: faster and more sensitive homology search. *Bioinformatics* **18** 440-445.
32. Morgenstern, B. (1999) Dialign2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, **15**, 211-218.
33. Notredame, C. & Higgins, D. G. (1996) SAGA: sequence alignment by genetic algorithm. *Nucl. Acids Res.* **24**, 1515-1524.
34. Notredame, C., Higgins DG, Heringa J. (2000) T-Coffee: a novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.* 2000, **302**, 205-217.
35. Notredame,C. (2002) Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics*, **3**, 131-144.

36. Roditty, L. and Zwick, U. (2002) Improved dynamic reachability algorithms for directed graphs. *Proceedings of FOCS'02*, 679-689.
37. Roditty, L. and Zwick, U. (2004) A fully dynamic reachability algorithm for directed graphs with an almost linear update time. *Proc. of 36th STOC*, 184-191.
38. Saitou, N. & Nei, M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.* **4**, 406-125.
39. Stoye, J., Moulton, V. & Dress, A. W. (1997) DCA: an efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment. *Comput. Appl. Biosci.* **13**, 625-626
40. Sun, Y., Buhler, J. (2004) Designing multiple simultaneous seeds for DNA similarity search. *Proc. 8th Annual International Conference on Computational Molecular Biology (RECOMB)*. 76-84
41. Thompson, J., Higgins, D. & Gibson, T. (1994) ClustalW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Acids Res.* **22**, 4673-4690
42. Thompson, J., Plewniak, F. & Poch, O. (1999) BaliBase: a benchmark alignment database for the evaluation of multiple sequence alignment programs. *Bioinformatics*, **15**, 87-88.
43. Thompson, J. D., Plewniak, F. & Poch, O. (1999) A comprehensive comparison of multiple sequence alignment programs. *Nucl. Acids Res.* **27**, 2682-2690.
44. Wang, L. & Jiang, T. (1994) On the complexity of multiple sequence alignment. *J. Comput. Biol.* **1**, 337-348.