ELSEVIER

Theoretical Computer Science 182 (1997) 233–244

Theoretical Computer Science

Note

Approximation algorithms for multiple sequence alignment

Vineet Bafna[a,1], Eugene L. Lawler[b,2], Pavel A. Pevzner[c,*]

[a] *Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802, USA*

[b] *Computer Science Division, University of California, Berkeley, CA 94720, USA*

[c] *Department of Mathematics and Computer Science, University of Southern California, Los Angeles, CA 90089-1113, USA*

Received January 1995; revised February 1996

Communicated by M. Crochemore

Dedicated to the memory of Eugene Lawler

Abstract

We consider the problem of aligning of k sequences of length n. The cost function is sum of pairs, and satisfies triangle inequality. Earlier results on finding approximation algorithms for this problem are due to Gusfield (1991) who achieved an approximation ratio of $2 - 2/k$, and Pevzner (1992) who improved it to $2 - 3/k$. We generalize this approach to assemble an alignment of k sequences from optimally aligned subsets of $l < k$ sequences to obtain an improved performance guarantee. For arbitrary $l < k$, we devise deterministic and randomized algorithms yielding performance guarantees of $2 - l/k$. For fixed l, the running times of these algorithms are polynomial in n and k.

1. Introduction

Multiple sequence alignment is a fundamental problem in computational molecular biology. Alignments of multiple sequences are commonly computed for the purpose of discovering 'homologous', i.e., evolutionarily or functionally related, regions of the sequences. An *optimal* multiple alignment can be computed by dynamic

* Corresponding author.

[1] The research was supported in part by the National Science Foundation Young Investigator Award, the National Science Foundation under grant CCR-9308567, the National Institute of Health under grant R01 HG00987 and the DOE grant DE-FG03-90ER60999.

[2] Deceased.

PII S0304-3975(97)00023-6

programming. However, the running time of dynamic programming algorithms increases rapidly with k, the number of sequences to be aligned. Accordingly, many heuristics and approximation algorithms have been proposed [1, 6, 11–13].

Many objective functions have been suggested for the multiple sequence alignment problem. One of the most widely used is the 'sum-of-pairs' (SP) criterion. The problem of computing an optimal alignment with respect to the sum-of-pairs criterion is NP-hard [20]. The advanced algorithms [12] allow one to construct *optimal* alignments of $k \leqslant 6$ sequences, each of length around 200, the length of an average protein. Many algorithms for k sequences use optimal multiple alignment of $l < k$ sequences with further assembling of these "partial" alignments into an *approximate* alignment of k sequences. Multiple alignment algorithms based on this heuristic are widely used un computational molecular biology [19], and are known to produce meaningful biological results [8]. This approach requires an efficient "assembly" procedure providing an approximate alignment of k sequences close to the optimal one. However, no 'performance guarantee' algorithms for multiple alignment have been known until recently, although a number of heuristics for *suboptimal* multiple alignment have been developed (see the recent review, [6]).

Gusfield [9, 10] achieved an approximation ratio of $2 - 2/k$ by assembling an alignment of k sequences from optimal alignments of pairs of sequences. It is known that models currently employed to align sequences are not quite adequate; thus, for practical sequence alignment it is not always necessary to produce an optimal alignment but only one that is plausible. The Gusfield algorithm produces plausible alignments; a computational experiment with an alignment of 19 sequences gave a suboptimal solution only 2% worse than the optimal one. An obvious direction for improvement is to use optimal alignments of $l > 2$ sequences, and then assemble them to approximately align k sequences. However, devising an efficient "assembling" procedure for an arbitrary l remained an open problem.

Pevzner [15] improved the performance guarantee to $2 - 3/k$ by assembling optimal alignments of triples of strings. This suggests the possibility of achieving a further improvement in the performance guarantee to $2 - l/k$ by assembling l-way alignments. We investigate this possibility, and show that for arbitrary $l < k$ it is possible to obtain such a performance guarantee with a running time that is polynomial in n and k. This result provides an evidence that "assembling of alignments" heuristics, commonly used in computational molecular biology [19] might give "good" suboptimal alignment if assembling is done carefully.

In Sections 2 and 3 we define SP-alignment formally, and outline a heuristic approach to constructing SP-alignments of k sequences by combining alignments of l sequences. In Section 4, we show that the problem of constructing SP-alignments within a desired performance ratio reduces to constructing *balanced* sets of *l-stars*. In Section 5, we use dynamic programming to get some improvement over the brute-force approach. Section 6 deals with constructing small balanced sets to ensure small running time. Finally, in Section 7, we show how to obtain an efficient randomized algorithm for SP-alignment.

2. Definitions

Let $\mathscr{A}$ be a finite *alphabet* and $a_1, \dots a_k$ be k sequences (strings) over $\mathscr{A}$. For convenience, we assume that each of these strings contains n characters. Let $\mathscr{A}'$ denote $\mathscr{A} \cup \{-\}$, where "–" denotes "space". An *alignment* of strings $a_1, \dots, a_k$ is specified by a $k \times m$ matrix A, where $m \geqslant n$. Each element of the matrix is a member of $\mathscr{A}'$, and each row i contains the characters of a_i in order, interspersed by $m-n$ spaces.

Given an alignment A we denote A_{ij} a pairwise alignment formed by the rows i and j of A. The score of an alignment is determined with reference to a symmetric matrix D specifiying the dissimilarity or *distance* between elements of $\mathscr{A}'$. We assume the metric properties for distance d, so that $d(x,x)=0$ and $d(x,z) \leqslant d(x,y)+d(y,z)$, for all x,y,z in $\mathscr{A}'$. For a given alignment $A=[a_{ih}]$, the *score* for sequences a_i, a_j is

$$s(A_{ij}) = \sum_{h=1}^{m} d(a_{ih}, a_{jh}),$$

and the *sum-of-pairs score* (SP-score) for the alignment A is given by $\sum_{i,j} s(A_{ij})$. In this definition the score of alignment A is the sum of the scores of *projections* of A onto all pairs of sequences a_i and a_j. Let $C=[c_{ij}]$ be a $k \times k$ matrix of *weights* where c_{ij} is the "weight" of the pairwise alignment between a_i and a_j. The *weighted sum-of-pairs score* for the alignment A is

$$\sum_{i,j} c_{ij} s(A_{ij}).$$

For notational convenience we use *matrix dot product* to denote scores of alignments. Thus, letting $S(A)=[s(A_{ij})]$ be the matrix of scores of pairs of sequences, the weighted sum-of-pairs score is $C\,S(A)$. Letting E be the unit matrix consisting of all 1's except the main diagonal consisting of all 0's, the (unweighted) sum-of-pairs score of alignment A is $E\,S(A)$.

Straightforward dynamic programming, with running time $O((2n)^k)$, solves the weighted sum of pairs alignment problem for k sequences. A number of different variations, and some speedups of the basic algorithm have been devised [16, 17, 21]. Hereafter, we let $g(k,n)$ denote the running time required to obtain an optimal solution to the weighted sum-of-pairs problem for k sequences of length n.

3. Compatible alignments

Given an alignment A on sequences $a_1, \dots, a_k$ and an alignment A' on some subset of the sequences, we say that A is *compatible* with A' if A aligns the characters of the sequences aligned by A' in the same way that A' aligns them. Feng and Doolittle [7] observed that given any tree in which each vertex is labeled with a distinct sequence a_i, and pairwise alignments specified for each tree edge, there exists an alignment of the k sequences that is compatible with each of the pairwise alignments. A similar result holds for "l-stars", defined as follows:

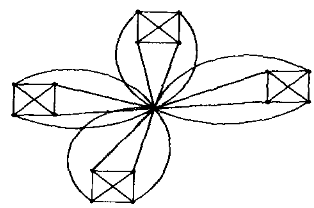

Fig. 1. A 5-star on 17 vertices.

Let V be the set $\{1,2,\dots,k\}$ representing the sequences $a_1,a_2,\dots,a_k$, and suppose $l-1|k-1$. An *l-star* $G=(V,E)$ is defined by $r=(k-1)/(l-1)$ cliques of size l whose vertex sets intersect in only one *center* vertex (Fig. 1). Let $A_1\dots,A_r$, be alignments for the r cliques, with each A_i aligning l sequences. By a construction similar to Feng and Doolittle [7] we have the following lemma:

Lemma 1. *For any l-star and any specified alignments $A_1,\dots,A_r$ for its cliques, there is an alignment A for the k sequences that is compatible with each of the alignments $A_1,\dots,A_r$.*

Proof. Assume that the alignment A_i $(1\leqslant i\leqslant r)$ is specified by an $l\times m_i$ matrix with the first row corresponding to the center vertex (string) a_1. We transform matrices $A_1,\dots,A_r$ into $l\times m^*$ matrices $A_1^*,\dots,A_r^*$ by "padding" m^*-m_i columns consisting of spaces into A_i for $1\leqslant i\leqslant r$, as follows.

Let $m\leqslant m_i$ be the length of the center string a_1. A_i contains m symbols from a_1 and m_i-m space symbols in the first row. Let $j_{i,1},\dots,j_{i,m}$ be the positions of m symbols from a_1 in the first row of A_i. Denote $z_{i,l}=j_{i,l+1}-j_{i,l}$ the number of space symbols between lth and $(l+1)$th non-space symbols in the first row of A_i (we assume $z_{i,0}=1$ and $z_{i,m+1}=m_i$). Clearly, $\sum_{0\leqslant l\leqslant m} z_{i,l}=m_i-m$.

Let $z_l=\max_{1\leqslant i\leqslant r} z_{i,l}$ be the maximum spacing between lth and $(l+1)$th non-space symbols in the first row of the matrices $A_1,\dots,A_r$. Denote $m^*=m+\sum_{l=0}^{m} z_l$ and transform $l\times m_i$ matrix A_i into $l\times m^*$ matrix A_i^* by adding $z_l-z_{i,l}$ "space" columns (i.e. columns consisting of space symbols) between the columns j_l and j_{l+1} of A_i. Matrices $A_1^*,\dots,A_r^*$ have the same number of columns and union of their rows generates an alignment A compatible with the alignments $A_1,\dots,A_r$ (see [7] for more details). $\square$

Assign weights to the edges of an l-star G, with center c, as follows.

$$c_{ij}=\begin{cases} k-(l-1) & i=c \text{ or } j=c,\\ 1 & i,j\neq c,\ i \text{ and } j \text{ are contained in the same clique of } G,\\ 0 & \text{otherwise}\end{cases}$$

and let $C(G)=[c_{ij}]$ denote the $k \times k$ matrix of weights. Note that

$$C(G)E=(k-(l-1))\,(k-1)+\binom{k-1}{l-1}\binom{l-1}{2}=\binom{k}{2}\left(2-\frac{l}{k}\right).$$

The pairwise scores of an alignment inherit the triangle inequality property from the distance matrix D. That is, for any alignment A, $s(A_{ij}) \leqslant s(A_{ik})+s(A_{kj})$, for all i,j,k. This fact was used by Pevzner [15] to prove the following:

Lemma 2. *For any alignment A of the k sequences, and an l-star G, $E\,S(A) \leqslant C(G)\,S(A)$.*

Let $C_1,\ldots,C_r$ denote the submatrices of weights for the r cliques of an l-star G. Let $A_1^*,\ldots,A_r^*$ be optimal weighted sum-of-pairs alignments for the r cliques. From Lemma 1 and the fact that $d(-,-)=0$, we obtain the following.

Lemma 3. *Given an l-star G, there is an optimal (weighted with respect to $C(G)$) alignment A_G for the k sequences that is compatible with each of the alignments $A_1^*,\ldots,A_r^*$. Moreover, $C(G)S(A_G)=C_1\,S(A_1^*)+\ldots+C_r\,S(A_r^*)$.*

To summarize, for any l-star G we can assemble an alignment A_G, optimal with respect to the weight matrix $C(G)$ specified above, by computing optimal weighted alignments for each clique of G. This can be done in $O(kg(l,n))$ time.

4. Balanced sets of l-stars

Let $\mathscr{G}$ be a collection of l-stars, and let $C(G)$ denote the weight matrix for star G. We say that the collection $\mathscr{G}$ is *balanced* if $\sum_{G\in\mathscr{G}} C(G)=pE$ for some scalar $p>1$.

Lemma 4. *If $\mathscr{G}$ is a balanced set of l-stars, then*

$$\min_{G\in\mathscr{G}} C(G)S(A_G) \leqslant \frac{p}{|\mathscr{G}|}\min_A E\,S(A).$$

Proof. We use an averaging argument.

$$\begin{aligned}\min_{G\in\mathscr{G}} C(G)S(A_G) &\leqslant \frac{1}{|\mathscr{G}|}\sum_{G\in\mathscr{G}} C(G)S(A_G)\\ &\leqslant \frac{1}{|\mathscr{G}|}S(A)\sum_{G\in\mathscr{G}} C(G)=\frac{p}{|\mathscr{G}|}E\,S(A).\end{aligned}$$

Here the inequality holds for an arbitrary alignment A, and in particular, it also holds for the optimum alignment. □

Lemmas 2 and 4 motivate the algorithm *Align* (Fig. 2).

Procedure *Align*

1. Construct a balanced set of l-stars, $\mathscr{G}$.
2. For each l-star G in $\mathscr{G}$, assemble an alignment A_G that is optimal with respect to $C(G)$ from alignments that are optimal for each of its cliques (Lemma 3).
3. Choose G with the corresponding alignment A_G such that $C(G) \cdot S(A_G)$ is the minimum over all l-stars in $\mathscr{G}$. Return A_G.

Fig. 2. Deterministic algorithm for multiple alignment.

Theorem 1. *Given a balanced collection of l-stars $\mathscr{G}$, Align returns an alignment with a performance guarantee of $2 - l/k$ in $O(k|\mathscr{G}|g(l,n))$ time.*

Proof. Note that

$$\frac{p}{|\mathscr{G}|} = \frac{C(G)E}{EE} = 2 - \frac{l}{k}.$$

Now, *Align* returns the alignment A_G which is optimal for l-star $G \in \mathscr{G}$, and for which the smallest weighted score, $\min_{G \in \mathscr{G}} C(G)S(A_G)$ is achieved. Lemmas 2 and 4 imply that $ES(A_G) \leqslant C(G)S(A_G) \leqslant (2 - \frac{l}{k}) \min_A E\,S(A)$. □

5. Optimizing over all l-stars

We have reduced our approximation problem to that of finding an optimal alignment for each l-star in a balanced set. How hard is it to find a balanced set $\mathscr{G}$? A trivial candidate is simply the set of all l-stars, which is clearly balanced by symmetry. Note that for $l = 2$, there are only k l-stars. This fact was exploited by Gusfield [9] to obtain an approximation ratio of $2 - 2/k$. This is really a special case, as for $l > 2$, the number of l-stars grows exponentially with k making the algorithm computationally infeasible. Pevzner [15] solved the case of $l = 3$, by mapping the problem to weighted matching on graphs.

In this section, we show that it is not necessary to exhaustively compute alignments for all possible l-stars. Dynamic programming provides a shortcut. Specifically, we prove the following:

Theorem 2. *For all k, l, it is possible to compute an alignment with a performance guarantee of $2 - l/k$ in $O(k^{l+1}(2^k + kg(l,n)))$ time.*

Proof. For simplicity, consider at first the case when $l-1 \mid k-1$. Fix a center vertex c. Consider an arbitrary subset Q of $l-1$ sequences from $V \setminus c$. Denote $opt(Q)$ to be the optimum score of a weighted alignment of the sequences in Q along with c such that the weight of all edges incident to c is $k-l+1$ and the weight of the remaining edges is 1. For each choice of a center vertex c and for each of the $\binom{k-1}{l-1}$ possible cliques $Q \subseteq V \setminus c$, compute $opt(Q)$. This computation can be done in time $O(k\,k^l\,g(l,n))$.

Next, for all $Q \subseteq V \backslash c$, such that $|Q|$ is a multiple of $l-1$, denote $s(Q)$ as the minimum alignment score among all l-stars over the vertices in $Q \cup c$ with center vertex c Now, let $Q_1, Q_2 \ldots, Q_r$ be the cliques on an l-star with the minimum alignment score. Then, from Lemma 3, $s(Q) = opt(Q_1) + opt(Q_2) + \cdots + opt(Q_r)$. Clearly, $s(Q)$ can be computed by the following recurrence:

$$s(\phi) = 0$$

$$s(Q) = \min_{Q' \subseteq Q, |Q'| = l-1} \{s(Q \setminus Q') + opt(Q')\}$$

Q is a set of size at most $k-1$. In order to compute $s(Q)$, we need to look at most $\binom{k-1}{l-1} = O(k^l)$ subsets Q'. Therefore, computing $s(Q)$ for each of at most 2^k sets Q takes $O(k^l)$ time, and repeating for each choice of a center vertex, the computation takes $O(k^l\,2^k\,k)$ time. Therefore, if $l-1|k-1$, we can compute the optimum score in $O(k^l(2^k + kg(l,n)))$ time.

In the general case, when $(l-1)$ does not divide $(k-1)$, we need to consider *hybrid* stars which contain cliques of size l as well as $l+1$. Therefore, we compute $opt(Q)$ for all cliques of size l or $l+1$ in time $O(k^{l+1}kg(l+1,n))$. The new recurrence for $s(Q)$ is as follows:

$$s(\phi) = 0$$

$$s(Q) = \min_{Q' \subseteq Q, |Q'| = l-1 \text{ or } |Q'| = l} \{s(Q \backslash Q') + opt(Q')\}.$$

The net running time increases to $O(k^{l+1}(2^k + kg(l+1,n)))$. □

This approach may be computationally tractable for many problem instances. However, in order to obtain a time bound that is polynomial in n and k, for fixed l, we need to construct balanced sets of l-stars of small size.

Constructing a *small* balanced set of l-stars is not trivial, except for some specific values of l and k. One way of constructing such a set $\mathscr{G}$ for specific values of l and k is to consider a *sharply doubly transitive* set of permutations, and combinatorial block designs [2].

In the following section, we get around the difficulty of constructing small balanced sets *for all* l, k by constructing a balanced set that is exponentially large, but on which we can quickly find a minimum score l-star by solving matching problems.

6. Balanced sets of $(2l-1)$-stars

In this section, we prove the following theorem

Theorem 3. *For all* k, l, *it is possible to compute an alignment with a performance guarantee of* $2 - l/k$, *in* $O(k^3 g(2l+5, n))$ *time.*

Proof. For simplicity, let us first assume that $2(l-1) \mid k-1$. For each choice of a center vertex c, let G be an arbitrary l-star with r cliques. Define a *configuration* G' by combining the cliques of G in a pairwise fashion (to form $(2l-1)$-cliques), and assigning weights as follows:

$$c_{ij} = \begin{cases} k-(l-1)-1/2 & i=c \text{ or } j=c, \\ 1 & i,j \neq c,\ i \text{ and } j \text{ are contained in the same clique} \\ & \quad \text{of } G', \text{ but different cliques of } G, \\ 0 & \text{otherwise.} \end{cases}$$

Note that, as in the case of l-stars,

$$C(G)E = (k-1)\left[k-(l-1)-\frac{1}{2}\right] + \frac{k-1}{2(l-1)}(l-1)^2 = \binom{k}{2}\left(2-\frac{l}{k}\right).$$

Trivially, Lemmas 2 and 3 hold for a configuration also.

For an arbitrary l-star G with center c, consider the set of all configurations obtained by pairing up cliques in G. Consider an arbitrary edge (i,j) such that $i,j \neq c$, and i,j do not belong to the same clique of G. By symmetry, each such edge will appear an equal number of times, say x, in the set of all configurations.

Now, for each l-star G in a set of k arbitrary l-stars, each with a different center vertex, consider the set of all configurations obtained by pairing up cliques in G. We assert that this set of configurations, along with x copies of each l-stars, forms a balanced set $\mathscr{G}$. For an arbitrary entry in $C(G)$, $C(G)[i,j] = k-(l-1)-1/2$ exactly $(2/k)|\mathscr{G}|$ times (when i or j is the center vertex of G), and $c(G)[i,j]=1$ exactly $(k-2)/kx$ times. Therefore $\sum_{G\in\mathscr{G}} C(G) = pE$, where

$$p = \frac{(\sum_{G\in\mathscr{G}} C(G))E}{\binom{k}{2}}$$

is a scalar. Furthermore, $\sum_{i,j} C(G)[i,j] = (2-l/k)\binom{k}{2}$ is the same for all $G \in \mathscr{G}$, implying that

$$\left(\sum_{G\in\mathscr{G}} C(G)\right)E = \sum_{G\in\mathscr{G}} (C(G)E) = \sum_{G\in\mathscr{G}} (2-l/k)\binom{k}{2}.$$

Therefore, $p = (2-l/k)|\mathscr{G}|$.

Next, we show that we can compute the optimal weighted cost configuration without explicitly generating the set of all configurations. Fix k arbitrary l-stars, one for each choice of a center vertex. For each l-star with center c, form a complete graph of r vertices H_r, with each node corresponding to a clique of the l-star and the weight of an edge being the cost of an optimal weighted alignment on the corresponding $(2l-1)$-clique.

Note that each configuration of G with center c describes a matching in H_r. Further, by Lemma 3 the cost of the configuration is equal to the sum of weights on the matching edges. Therefore, a minimum cost matching on H_r gives the cost of an optimal weighted configuration of G with center c. In order to find the optimal weighted cost configuration in $\mathscr{G}$, we solve the corresponding matching problem for each choice of a center vertex and pick one with the minimum cost. Finally compare the optimal configuration with each cost of the k l-stars, and return one with the minimum cost. From earlier arguments, the corresponding alignment achieves the desired performance ratio.

For the running time, observe that in computing the graph H_r, we need to solve $\binom{r}{2}$ alignments of $2l-1$ sequences, which takes time $O(r^2 g(2l-1,n)) = O(k^2 g(2l-1,n))$. For typical values of n, k, this dominates the cost of computing a minimum cost matching on a graph of size r. Repeating this for each choice of a center vertex takes time $O(k^3 g(2l-1,n))$. Finally, computing the alignment for each of the k l-stars takes time $O(kg(l,n))$. Therefore, if $2(l-1)|k-1$, it is possible to compute an alignment with performance guarantee of $2 - l/k$, in $O(k^3 g(2l-1,n))$ time.

This method can be generalized for arbitrary l with a slight increase in running time. Consider a *hybrid* l-star G with an even number of cliques of size l and $l+1$. As before, define a configuration G' by combining cliques of G arbitrarily in a pairwise fashion to form new cliques of sizes $2l-1, 2l$ and $2l+1$. Assign weights exactly as before. Note that Lemmas 2 and 3 still hold. Also, from symmetry, if we take the set of all configurations of l-star G, then each edge that does not belong to a clique of G will appear an equal number of times, say x. Combining this with x copies of G, each edge appears exactly x times. By earlier arguments, this set is also balanced. The only thing that remains is to estimate the value of p. Note that,

$$C(G)E \leqslant (k-1)\left[k-(l-1)-\frac{1}{2}\right] + \frac{k-1}{2(l-1)}l^2 \leqslant \binom{k}{2}\left(2-\frac{l-2}{k}\right).$$

Therefore, $p \leqslant (2-(l-2)/k)$. Repeating earlier arguments, we see that the optimal weighted cost configuration for each choice of a center can be computed in time $O(k^2 g(2l+1,n))$. Therefore, in time $O(k^3 g(2l+1,n))$, we can compute an alignment that will guarantee a performance of $2-(l-2)/k$. For $l' = l-2$, this implies an algorithm that runs in time $O(k^3 g(2l'+5,n))$ and guarantees a performance of $2 - l'/k$. $\square$

As an aside, a smaller balanced set can be explicitly constructed. Let

$$r = \left\lfloor \frac{k-1}{2(l-1)} \right\rfloor.$$

A *perfect matching* on H_{2r} corresponds to a configuration in the original graph. It is easy to see that a set of configurations corresponding to a *1-factorization* of H_{2r} (edge-disjoint decomposition of H_{2r} into perfect matchings), for each of the k l-stars, along with a single copy of each l-star, forms a balanced set of size $O(k^2)$.

7. Random sampling of l-stars

What is the performance bound if we choose an l-star at random? Gusfield studied this for 2-stars and gave a bound on the expected score of the alignment [12]. Assuming a uniform distribution on the set of all l-stars, we are interested in the expected value of the random variable $C(G)S(A_G)$. As the set of all l-stars is balanced, $Exp[C(G)S(A_G)] \leqslant (2 - l/k)\min_A ES(A)$. However, it is not clear if we can pick with high probability, an l-star that achieves the $2 - l/k$ performance.

Let $\mathscr{G}_c$ be the set of all l-stars, with a fixed center c. For G in $\mathscr{G}_c$, let $C(G) = C_1(G) + C_2(G)$ be the partition of weight matrix into *Border and Center* weights, with $C_1(G)$ being the same as $C(G)$ except for the cth row and column which are 0. Define $E = E_1 + E_2$ in an identical manner. Observe the balancing property of $C_1(G)$, i.e. $\sum_{G\in\mathscr{G}_c} C_1(G) = p_1 E_1$, where

$$p_1 = \frac{(l-2)}{(k-2)}|\mathscr{G}_c|.$$

We have the following lemma:

Lemma 5. *For G chosen uniformly at random from $\mathscr{G}_c$, and any alignment A,*

$$Prob\left[C_1(G)S(A) > \frac{3}{2}\frac{p_1}{|\mathscr{G}_c|}E_1 S(A)\right] < \frac{2}{3}.$$

Proof. Let $BAD = \{G \in \mathscr{G}_c | C_1(G)S(A) > \frac{3}{2}\frac{p_1}{|\mathscr{G}_c|}E_1S(A)\}$. Then,

$$\begin{aligned}\frac{3}{2}\frac{p_1}{|\mathscr{G}_c|}E_1S(A)|BAD| &< \sum_{G\in BAD} C_1(G)S(A) \leqslant \sum_{G\in\mathscr{G}_c} C_1(G)S(A)\\ &= p_1E_1S(A)\end{aligned}$$

which implies that $|BAD| \leqslant \frac{2}{3}|\mathscr{G}_c|$. $\square$

Pick m l-stars randomly from $\mathscr{G}_c$. It follows from the proof of Lemma 5 that the l-star with the minimum weight alignment (among these m stars) is in BAD with probability less than or equal to $(\frac{2}{3})^m$. *Randomized_Alignment* (Fig. 3) uses this fact to construct a set of l-stars which guarantees a good performance with high probability.

Theorem 4. *If $l - 1 | k - 1$, then for an arbitrary $\varepsilon > 0$, Randomized_Alignment runs in time $O(k^2\lceil \lg(k/\varepsilon)\rceil g(2l, n))$, and returns an alignment that, with probability $1 - \varepsilon$, achieves a performance bound of $2 - l/k$.*

Proof. Consider the set of l-stars in $\mathscr{G} = \{G_c\colon\ 1 \leqslant c \leqslant k\}$, constructed by the outer loop. To begin with, assume that none of the l-stars in $\mathscr{G}$ is in BAD. In other words,

$$\text{for all } G \in \mathscr{G},\quad C_1(G)S(A) \leqslant \frac{3}{2}\frac{p_1}{|\mathscr{G}_c|}E_1S(A).$$

Procedure *Randomized_Alignment*(l, k, ε)
$\mathcal{G} \leftarrow \emptyset$
for $c \in \{1, \dots, k\}$
  repeat $2\lceil \lg(k/\varepsilon) \rceil$ times
    choose a random l-star G with center c
    compute an alignment A_G with the minimum weighted score $\min_A C(G)S(A)$
  $G_c \leftarrow$ an l-star with minimum weighted score among the $2\lceil \lg(k/\varepsilon) \rceil$ l-stars,
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{G_c\}$.
$G \leftarrow$ an l-star with the minimum weighted score $\min_{G \in \mathcal{G}} C(G)S(A_G)$
return A_G

Fig. 3. Randomized algorithm for multiple alignment.

Randomized_Alignment returns an l-star G with the minimum weighted score from $\mathcal{G}$. We give a bound on its score by a counting argument. For every alignment A,

$$\begin{aligned}
\min_{G \in \mathcal{G}} C(G)S(A_G) &\leqslant \min_{G \in \mathcal{G}} C(G)S(A) \leqslant \frac{1}{k} \sum_{G \in \mathcal{G}} C(G)S(A) \\
&= \frac{1}{k} \sum_{G \in \mathcal{G}} C_2(G)S(A) + \frac{1}{k} \sum_{G \in \mathcal{G}} C_1(G)S(A) \\
&\leqslant \frac{2}{k}(k - (l-1))ES(A) + \frac{k-2}{k} \frac{3}{2} \frac{l-2}{k-2} ES(A) \\
&= \left(2 - \frac{l}{2k}\right) ES(A).
\end{aligned}$$

Now, recall from Lemma 2 that $ES(A_G) \leqslant C(G)S(A_G)$, which implies that if none of the l-stars in $\mathcal{G}$ is in BAD, the algorithm achieves a performance bound of $(2 - l/2k)$.

Next, we show that none of the l-stars in $\mathcal{G}$ is in BAD with high probability. In each iteration of the inner loop, we consider $2\lceil \lg(k/\varepsilon) \rceil$ random l-stars, and pick a G_c with the minimum weighted score. By definition, this l-star is in BAD only if each l-star picked in that iteration is in BAD. Therefore, for all $1 \leqslant c \leqslant k$, the probability that $G_c \in \mathcal{G}$ is in BAD is less than

$$\left(\frac{2}{3}\right)^{2\lceil \lg(\frac{k}{\varepsilon}) \rceil} < \frac{\varepsilon}{k}.$$

The probabilty that none of the k $G_c \in \mathcal{G}$ are in BAD is greater than or equal to $1 - \varepsilon$.

Now, choose $l' = l/2$. Note from Lemma 3 that computing each alignment A_G takes time $O(kg(2l', n))$. Therefore, *Randomized_Alignment* runs in time $O(k^2 2\lceil \lg(k/\varepsilon) \rceil g(2l', n))$ and returns an alignment that, with probability $1 - \varepsilon$, achieves a performance bound of $2 - l'/k$. $\square$

Acknowledgements

We are grateful to Piotr Berman, Dima Grigoriev, Jeanette Schmidt and Martin Vingron for many useful comments and suggestions.

References

[1] S.F. Altschul, D.J. Lipman, Trees, stars, and multiple biological sequence alignment, SIAM J. Appl. Math. 49 (1989) 197–209.
[2] V. Bafna, E.L. Lawler, P. Pevzner, Approximation Algorithms for Multiple Sequence Alignment, in: Proc. of the 5th Annual Symp. on Combin. Pattern Matching (CPM'94). Lecture Notes in Computer Science, vol. 807, Springer, Berlin, 1994, pp. 43–53.
[3] Z. Baranyai, On the factorization of the complete uniform hypergraph, in: A. Hajnal, T. Rado, V.T. Sós (Eds.), Infinite and Finite Sets, North-Holland, Amsterdam, 1975, pp. 91–108.
[4] J. Bósak, Decompositions of Graphs, Kluwer, Dordrecht, 1990.
[5] J.L. Carter, M.N. Wegman, Universal classes of hash functions, J. Comput. System Sci. 18 (1979) 143–154.
[6] S.C. Chan, A.K.C. Wong, D.K.Y. Chiu, A survey of multiple sequence comparison methods, Bull. Math. Biol. 54 (1992) 563–598.
[7] D. Feng, R. Doolittle, Progressive sequence alignment as a prerequisite to correct phylogenetic trees, J. Molec. Evol. 25 (1987) 351–360.
[8] A.E. Gorbalenya, V.M. Blinov, A.P. Donchenko, E.V. Koonin, An NTP-binding motif is the most conserved sequence in a highly diverged monophyletic group of proteins involved in positive strand RNA viral replication. J. Molec. Evol. 28 (1989) 256–68.
[9] D. Gusfield, Efficient methods for multiple sequence alignment with guaranteed error bounds, Tech. Report, Computer Science Division, University of California, Davis, CSE-91-4, 1991.
[10] D. Gusfield, Efficient methods for multiple sequence alignment with guaranteed error bounds, Bulletin of Mathematical Biology 55 (1993) 141–154.
[11] T. Jiang, E.L. Lawler, L. Wang, Aligning sequences via an evolutionary tree: complexity and approximation, in: Proc. ACM STOC'94, 1994, pp. 760–769.
[12] J. Kececioglu, The maximum weight trace alignment problem in multiple sequence alignment, in: A. Apostolico, M. Crochemore, Z. Galil, U. Manber (Eds.), Combinatorial Pattern Matching 93, Padova, Italy, June 1993, Lecture Notes in Computer Science, vol. 684, Springer, Berlin, pp. 106–119.
[13] D.J. Lipman, S.F. Altschul, J.D. Kececioglu, A tool for multiple sequence alignment, Proc. Nat. Acad. Sci. U.S.A 86 (1989) 4412–4415.
[14] P. Lorimer, Finite projective planes and sharply 2-transitive subsets of finite groups, in: Proc. 2nd Internat. Conf. Theory of Groups, Canberra (1973) 432–436.
[15] P. Pevzner, Multiple alignment, communication cost, and graph matching, SIAM J. Appl. Math. 52 (1992) 1763–1779.
[16] D. Sankoff, Minimum mutation tree of sequences, SIAM J. Appl. Math. 28 (1975) 35–42.
[17] D. Sankoff, Simultaneous solution of the RNA folding, alignment and protosequence problems, SIAM J. Appl. Math. 45 (1985) 810–825.
[18] J. Schmidt, A. Siegel, The analysis closed hashing under limited randomness, Proc. 22nd ACM Symp. on Theory of Computing, 1990, pp. 224–234.
[19] W.R. Taylor, Hierarchical method to align large numbers of biological sequences. Methods Enzymol. 183 (1990) 456–474.
[20] L. Wang, T. Jiang, On the complexity of multiple sequence alignment, J. Comp. Biol. 1 (1994) 337–348.
[21] M.S. Waterman, T.F. Smith, W.A. Beyer, Some biological sequence metrics. Adv. in Math. 20 (1976) 367–387.