BIOINFORMATICS

Vol. 24 ECCB 2008, pages i187–i192
doi:10.1093/bioinformatics/btn281

Segment-based multiple sequence alignment

Tobias Rausch[1,*], Anne-Katrin Emde[2], David Weese[2], Andreas Döring[2], Cedric Notredame[3] and Knut Reinert[2]

[1]International Max Planck Research School for Computational Biology and Scientific Computing, Ihnestr. 63-73, 14195 Berlin, Germany, [2]Algorithmische Bioinformatik, Institut für Informatik, Takustr. 9, 14195 Berlin, Germany and [3]Comparative Bioinformatics Group, Center for Genomic Regulation, Dr Aiguader 88, 08003 Barcelona, Spain

ABSTRACT

Motivation: Many multiple sequence alignment tools have been developed in the past, progressing either in *speed* or *alignment accuracy*. Given the importance and wide-spread use of alignment tools, progress in both categories is a contribution to the community and has driven research in the field so far.

Results: We introduce a graph-based extension to the consistency-based, progressive alignment strategy. We apply the consistency notion to segments instead of single characters. The main problem we solve in this context is to define segments of the sequences in such a way that a graph-based alignment is possible. We implemented the algorithm using the SeqAn library and report results on amino acid and DNA sequences. The benefit of our approach is threefold: (1) sequences with conserved blocks can be rapidly aligned, (2) the implementation is conceptually easy, generic and fast and (3) the consistency idea can be extended to align *multiple* genomic sequences.

Availability: The segment-based multiple sequence alignment tool can be downloaded from http://www.seqan.de/projects/msa.html. A novel version of T-Coffee interfaced with the tool is available from http://www.tcoffee.org. The usage of the tool is described in both documentations.

Contact: rausch@inf.fu-berlin.de

1 INTRODUCTION

Many bioinformatics applications rely on accurate multiple alignments. These include phylogenetic reconstruction, structure prediction, homology modeling, and consensus computation in assembly projects to name just a few. The computation of accurate multiple alignments is, however, a non-trivial task. The two main reasons are (1) that our current understanding of biology makes it hard to precisely define the mathematical properties of a biologically optimal alignment and (2) that even under highly simplified formulations (i.e. alignment score maximization) the problem is known to be NP-Hard (Wang and Jiang, 1994). Using standard scoring systems, dynamic programming algorithms (Carrillo and Lipman, 1988; Lermen and Reinert, 2000; Lipman *et al.*, 1989; Reinert *et al.*, 2000) or techniques from combinatorial optimization (Althaus *et al.*, 2002, 2006; Lenhof *et al.*, 1999) can be applied to compute an exact solution. This has certainly some benefits, but given the NP-hardness of the problem, it is only practical for alignments involving a few, relatively short sequences.

Hence, in practice, heuristic methods are used. Most algorithms use the progressive alignment paradigm (Feng and Doolittle, 1987) including Clustal W (Thompson *et al.*, 1994), T-Coffee (Notredame *et al.*, 2000), MUSCLE (Edgar, 2004), and many others. The progressive approach has two steps. First, using k-mer counts or pairwise alignment information, an initial guide tree is built and second, the sequences are incorporated one by one into the final alignment taking into account the order defined by the tree. During the progressive alignment phase some tools use the *consistency* concept (Gotoh, 1990). Consistency refers to a means of estimating whether the alignment of two given residues is consistent with the overall final alignment, or with any suitable collection of pre-computed pairwise or multiple alignments. In practice, this consistency is estimated by quantifying how many intermediate alignments (triplets) support the alignment of the two considered residues. The value computed in this way is used afterwards as a local substitution matrix. This strategy ensures the incorporation of as much sequence information as possible even in the early stages of the progressive alignment to avoid the inherent pitfalls of a greedy process. The combination of this consistency measure with a progressive alignment has been pioneered in the T-Coffee package (Notredame *et al.*, 2000), and later re-implemented and improved in a number of other tools.

We propose a segment-based, progressive alignment tool that uses consistency. The novelty of our approach is to work on sequence segments rather than on single characters and to provide a graph-based alignment of these segments that allows arbitrary input matches. We represent the segments as vertices in an alignment graph (Fig. 1). Meaningful segments can be defined using pairwise global and local alignments, maximal unique matches (MUMs) (Delcher *et al.*, 1999), or external pairwise comparison tools like BLAST (Altschul *et al.*, 1990). Unfortunately, direct collections of segments cannot readily be turned into an alignment graph, since some of the segments may overlap and intersect each other. A *refinement* step is therefore needed to split the segments so that in the final graph all segments are distinct from one another (Fig. 2). The pairwise refinement algorithm was described in (Halpern *et al.*, 2002) and used for a pairwise segment decomposition in (Szklarczyk and Heringa, 2006). In this article, we give the first extension of this procedure to multiple sequences and use this multiple segment match refinement algorithm in our graph-based alignment. The resulting algorithm is used by our two alignment tools, SeqAn::T-Coffee and SeqAn::M-Coffee, with improved speed and accuracy when compared to similar programs.

An important property of the algorithm is its ability to deal equally well with protein datasets and very long genomic sequences. We evaluated our implementation for proteins on two standard reference datasets: BAliBASE 3.0 (Thompson *et al.*, 2005, 1999) and PREFAB 4.0 (Edgar, 2004). For long sequences we used a set

*To whom correspondence should be addressed.

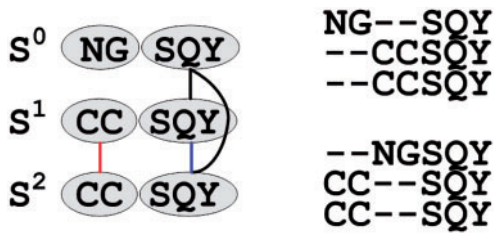

Fig. 1. Alignment graph: an alignment graph for 3 sequences S^0, S^1 and S^2. Vertices represent arbitrary segments, edges represent matches or mismatches, and gaps can be inferred from the topology. The order of adjacent indels is unspecified in an alignment graph. The blue edge between S^1 and S^2 connecting the SQY vertices is consistent with the intermediate sequence S^0 whereas the red edge connecting the CC vertices is not consistent.

Fig. 2. Segment match refinement: overlaps between segment matches are resolved using the refinement algorithm.

of six adenoviruses and a set of four different dengue serotypes causing the acute febrile disease dengue fever. We show that for amino acid sequences we are faster and more accurate than the previous M-Coffee (Wallace, *et al.*, 2006) and overall more accurate than the next best tools ProbCons (Chuong *et al.*, 2004) and MAFFT (Katoh *et al.*, 2002). For large, fairly similar DNA sequences, we show that we can quickly compute the most accurate large-scale alignments when compared to the other programs.

The article is structured as follows. We start in Section 2.1 by describing our graph-based model and the multiple alignment algorithm. This algorithm was derived from T-Coffee but operates on the segment-based alignment graph. In Section 2.2 we explain how to refine a given set of segment matches for multiple sequences. In Section 2.3 we explain how suitable sequence matches can be found for multiple amino acid or DNA alignments. Finally, we show results of our implementation in Section 3 and conclude with a discussion in Section 4.

2 METHODS

2.1 Graph-based multiple alignment

As shown in Figure 1, a multiple alignment $Align(S^0,S^1,S^2)$ of $n=3$ sequences S^0, S^1 and S^2 can be seen as an n-partite alignment graph $G=(V,E)$. Vertices in V represent sequence substrings (segments), edges in E represent matches or mismatches and gaps are implicitly represented by the topology of the graph. The last property implies that there is a one-to-many relationship between alignment graphs and alignment matrices because alignment graphs do not impose an order on adjacent indels. Alignment graphs can be converted to an alignment matrix using standard graph algorithms (Cormen *et al.*, 2001), namely connected components and topological sort. The former is used to group vertices connected by alignment edges into a single component, the latter ensures that the order of the components fits the order of the characters in the sequences. To minimize the size requirements of the graph, vertices store positions instead of sequence substrings. In practice, three integers are sufficient: (1) a sequence ID, (2) a starting position and (3) the length of the segment. The edges carry weights that indicate the benefit of aligning the segments connected by the edge.

Alignment graphs are, however, more general in the sense that they can contain a set of possible alignment edges E where only a subset $T \subset E$ constitutes a valid alignment. An example is shown in Figure 2 where only one edge incident to the CC vertex in S^0 can be realized by an alignment. The set of edges T that is realized by an alignment is called a trace (Kececioglu, 1993; Sankoff and Kruskal, 1983). The optimal alignment (*maximal trace*) corresponds to the trace where the sum of weights of all trace edges minus possible gap costs is maximal. The pairwise maximum trace can be computed: (1) using dynamic programming algorithms (Gotoh, 1982; Needleman and Wunsch, 1970) that allow arbitrary gap costs, or (2) using chaining algorithms (Abouelhoda and Ohlebusch, 2003; Myers and Miller, 1995) that allow simple gap cost functions or (3) using the heaviest common subsequence algorithm (Jacobson and Vo, 1992) that simply computes the heaviest trace without any gap costs.

In short, a single alignment graph contains match information about a set of sequences. Multiple alignment graphs can be combined by means of the segment match refinement algorithm described in the next section. The alignment graph itself is very similar to a T-Coffee *library*, except that in a library every segment is divided into single characters. Consistency is ensured by means of the triplet extension: (1) Traverse all vertices in the alignment graph and for each source vertex consider all possible pairs of adjacent vertices. (2) The two target vertices are either connected by an edge or disconnected. If they are connected, we have found a triplet and increase the weight of the connecting edge by the minimum weight of the other two edges. If they are not connected, we simply insert a new edge with minimum weight. Note that a progressive alignment on an alignment graph aligns two strings of vertices instead of sequence characters. A profile in terms of an alignment graph is a string where each position has a set of vertices. Thus, given a guide tree we can perform a progressive alignment on an alignment graph of multiple sequences in the same manner as aligning the sequences themselves. The approach is, however, more generic because a single vertex can be any group of characters, e.g. a large segment, a gene, or just a single character.

2.2 Multiple segment match refinement

To facilitate a graph-based alignment we extended the segment match refinement algorithm introduced in (Halpern *et al.*, 2002) to multiple sequences. Whereas a greedy method would have to choose between overlapping segment matches, the segment match refinement algorithm computes a *minimal* subdivision of the segments, i.e. it refines the segment matches such that all parts of all segment matches can be used (Fig. 2). The set of refined matches, as described below, is as small as possible to keep the alignment graph small.

Let $\mathcal{S}=\{S^0,S^1,...,S^{n-1}\}$ be a set of n sequences with $S^i=s^i_0 s^i_1 ... s^i_{|S^i|-1}$ and $i\in\{0,1,...,n-1\}$. Let $\mathcal{M}=\{M^0,M^1,...,M^{m-1}\}$ be a set of m segment matches with $M^k=(S^i_{uv},S^j_{xy})$ and $k\in\{0,1,...,m-1\}$. M^k is an alignment between two segments $S^i_{uv}=s^i_u s^i_{u+1}...s^i_{v-1}$ and $S^j_{xy}=s^j_x s^j_{x+1}...s^j_{y-1}$ with $i,j\in\{0,1,...,n-1\}$, $i\neq j$, $0\leq u<v\leq|S^i|$, and $0\leq x<y\leq|S^j|$. For the segment S^i_{uv} the positions u and v are the boundary positions. We define the *S^i-support* of $\mathcal{M}$, in short $supp_{S^i}(\mathcal{M})$, to be the set of all boundary positions of segments on sequence S^i.

Segment matches can be arbitrary alignments, i.e. gapped alignments. That is, the lengths of S^i_{uv} and S^j_{xy} can be different. In these cases, the algorithm requires projection maps to determine which character s^i_{u+l} on S^i_{uv} with $u+l\in\{u,u+1,...,v-1\}$ corresponds to a character s^j_{x+k} on S^j_{xy} with $x+k\in\{x,x+1,...,y-1\}$. If we exclude reversals we can, however, subdivide these alignments into exact segment matches where $v-u=y-x$ always holds. In this case, the projection of s^i_{u+l} is simply s^j_{x+l}.

The goal of the algorithm is to refine a set of input segment matches $\mathcal{M}$ into a set of segment submatches $\mathcal{M}_*=\{M^0_*,M^1_*,...,M^{m'-1}_*\}$ where all

submatches cover the original matches. A submatch of $M^k=(S^i_{uv},S^j_{xy})\in\mathcal{M}$ is a match $M^{k'}=(S^i_{u'v'},S^j_{x'y'})\in\mathcal{M}_*$ with $u\le u'<v'\le v$, $x\le x'<y'\le y$, $v'-u'=y'-x'$, and $u'-u=x'-x$. A set $\mathcal{M}_*$ is called a refinement of $\mathcal{M}$ if each $M^{k'}\in\mathcal{M}_*$ is a submatch of a $M^k\in\mathcal{M}$ and the set $\mathcal{M}_*$ tiles $\mathcal{M}$. That is, for each segment match $M^k=(S^i_{uv},S^j_{xy})\in\mathcal{M}$ we have a subset $\mathcal{M}'_*\subset\mathcal{M}_*$ where each $M^{k'}\in\mathcal{M}'_*$ is a submatch of M^k and the following two conditions are true:

$$[u,v-1]=\dot{\bigcup_{M^{k'}\in\mathcal{M}'_*}}[u',v'-1]$$

$$[x,y-1]=\dot{\bigcup_{M^{k'}\in\mathcal{M}'_*}}[x',y'-1]$$

In short, each original match must be tiled by submatches in $\mathcal{M}_*$.

We are, however, interested in a refinement $\mathcal{R}$ out of the set of possible refinements where all segments are either disjoint or identical, i.e. a refinement without partially overlapping segments. We call such a set of segment matches $\mathcal{R}$ *resolved*. In a resolved set any $(S^i_{uv},S^j_{xy})\in\mathcal{R}$ satisfies the requirement that $[u,v]\cap supp_{S^i}(\mathcal{R})=\{u,v\}$ and $[x,y]\cap supp_{S^j}(\mathcal{R})=\{x,y\}$. If we refine every segment match $M^k\in\mathcal{M}$ into single position matches we obtain a trivial resolved refinement. Hence, the objective is to find a refinement $\mathcal{R}$ of minimum cardinality.

Such a refinement can be constructed by the following algorithm that successively applies only the necessary cuts to resolve partial overlaps and terminates when all segments are disjoint or identical. The algorithm refines a set of input segment matches $\mathcal{M}$ and returns an alignment graph constructed from the refined segment match set $\mathcal{R}$. (1) The algorithm builds a node set V^i for each sequence $S^i\in\mathcal{S}$. Initially, $V^i=supp_{S^i}(\mathcal{M})$. (2) The segment matches are processed sequentially. For each segment match $M^k=(S^i_{uv},S^j_{xy})\in\mathcal{M}$ we extract the boundary positions u, v, x and y. (3) For each boundary position w we retrieve all segment matches $\mathcal{L}=\{L^0,L^1,...,L^{p-1}\}$ that contain w because these are the segment matches that partially overlap the given segment match. To perform this search efficiently we use an interval tree T^i for each sequence S^i. T^i stores all segments that the sequence S^i contains (Edelsbrunner, 1980). (4) We now sequentially process $\mathcal{L}$. Let w be the current boundary position then each $L^q=(S^i_{uv},S^j_{xy})$ with $q\in\{0,1,...,p-1\}$ and $u<w<v$ is used to project w onto S^j. Let h be this projected position. Then we have either $h\in V^j$ or $h\notin V^j$. If $h\in V^j$ the recursion stops. If $h\notin V^j$ we call h a cut and insert h into V^j. Then we recurse to (3) using h. (5) At the end of processing all segment matches each V^i contains all original boundary positions plus those positions added during the refinement. Note that no superfluous cuts are made since all positions in V^i are either original boundary positions or necessary cuts from projections of the former positions. Thus, the refinement is of minimum cardinality. (6) At the end of the algorithm each set V^i can be directly used to define segments on S^i in the alignment graph. The required edges can be derived from the original matches in $\mathcal{M}$.

2.3 Generating segment matches

The advantage of our method is that the input can be *any* set of segment matches. To illustrate it, we provide two different mechanisms to generate segment matches.

For protein alignments and short DNA sequences the standard match generation algorithm takes a set of sequences and builds all pairwise global and local alignments using the dynamic programming algorithms of Gotoh (Gotoh, 1982) and Waterman and Eggert (Waterman and Eggert, 1987). We compute the dynamic programming matrix column-wise and save the traceback pointers in a reduced alphabet that can be efficiently stored. This enables us to use these algorithms for fairly long sequences, e.g. a set of adenovirus genomes.

However, for very long (genomic) sequences algorithms using quadratic space are impossible to use and space-efficient dynamic programming algorithms [e.g. Hirschberg's algorithm (Hirschberg, 1975)] become too inefficient. For these problem instances we either use the enhanced suffix array implementation available in SeqAn to compute maximal unique matches or external tools such as BLAST (Altschul *et al.*, 1990). For fairly similar sequences, we can also generate segment matches using an all-against-all comparison of sequences by means of the longest common subsequence algorithm (Jacobson and Vo, 1992).

3 RESULTS

SeqAn (Döring *et al.*, 2008) is a library of different components for sequence analysis. Therefore, we implemented all parts of the tool described in this article as new building blocks within the library. We implemented basic graph types, graph algorithms, the match refinement algorithm, progressive alignment strategies and sequence analysis functions in SeqAn. Then we combined these components into two command-line tools, namely (1) SeqAn::M-Coffee and (2) SeqAn::T-Coffee.

3.1 Tools

SeqAn::M-Coffee is a multiple sequence alignment meta-method extending the original M-Coffee (Wallace, *et al.*, 2006). Using multiple sequence alignments from arbitrary aligners as input, it refines the highly consistent alignment matches and rapidly generates a new meta-alignment that is the most consistent with all input alignments. We use a protocol similar to M-Coffee but improved the efficiency by taking advantage of conserved blocks in the input alignments that can be represented by large segments in our graph approach. We configured SeqAn::M-Coffee and M-Coffee with different aligners, i.e. MAFFT (Katoh *et al.*, 2002), ProbCons (Chuong *et al.*, 2004), MUSCLE (Edgar, 2004), and our own method SeqAn::T-Coffee described below. SeqAn::M-Coffee and the original M-Coffee rely on T-Coffee to build the library. Therefore, we integrated our tool into T-Coffee so that it can be seamlessly called from the T-Coffee command-line. This also provides the user with all the convenient options available in T-Coffee. Hence, SeqAn::M-Coffee and M-Coffee solely differ in the extension of the library and the progressive alignment.

SeqAn::T-Coffee is our versatile multiple sequence alignment tool for amino acid and nucleotide sequences. It generates sequence matches depending on the input as described in Section 2.3. This set of matches is refined using the match refinement algorithm described in Section 2.2. The resulting alignment graph is extended using the triplet extension. Subsequently, a guide tree is built from pairwise distances using UPGMA (Sokal and Michener, 1958) or neighbor joining (Saitou and Nei, 1987). The progressive alignment is computed by means of the very efficient heaviest common subsequence algorithm (Jacobson and Vo, 1992).

3.2 Evaluation

We compared SeqAn::M-Coffee and SeqAn::T-Coffee for amino acid sequences with AMAP (Schwartz and Pachter, 2007), DIALIGN-T (Subramanian *et al.*, 2005), MAFFT (Katoh *et al.*, 2002), MUSCLE (Edgar, 2004), ProbCons (Chuong *et al.*, 2004), T-Coffee (Notredame *et al.*, 2000) and M-Coffee (Wallace, *et al.*, 2006) on the BAliBASE 3.0 (Thompson *et al.*, 2005, 1999) and PREFAB 4.0 (Edgar, 2004) benchmark alignment datasets. The results on BAliBASE are shown in Table 1. We used the standard reference sets RV11–RV50 of full-length sequences and

Table 1. BAliBASE 3.0: average sum-of-pairs scores (SP) and average total column scores (TC) for all reference sets RV11–RV50

Aligner	RV11 (38)		RV12 (44)		RV20 (41)		RV30 (30)		RV40 (49)		RV50 (16)		CPU
	SP	TC	SP	TC	SP	TC	SP	TC	SP	TC	SP	TC	Time (s)
AMAP	49.61**	25.50**	90.64**	77.75**	86.24**	27.27**	73.42**	33.10**	78.45**	39.67**	78.51*	42.44*	14928
DIALIGN-T	49.30**	25.32**	88.75**	72.55**	86.29**	29.20**	74.66**	34.90**	81.95**	45.22**	80.14**	44.25**	3843
MAFFT	67.11	44.61	93.63	83.75	92.67	45.27	85.55	56.93	91.97	59.69	90.00	56.19	2914
MUSCLE	59.30*	35.92*	91.70**	80.36**	89.21	35.15	80.27	38.27	86.74**	47.10**	85.65	48.69	1547
ProbCons	66.99	41.68	94.12	85.52	91.68	40.54	84.60	54.37	90.37	52.88	89.29	56.69	20480
SeqAn::T-Coffee	63.58	39.11	92.99	82.36	92.05	43.34	83.90	52.27	92.23	57.98	88.12	56.50	14596
T-Coffee	58.22**	31.34**	92.27	81.18	90.91	37.80	79.09*	36.57*	86.02**	48.20**	86.09	50.62	53970
M-Coffee [mu]	64.00	40.45	93.73	83.84	90.12	39.73	83.40	47.90	87.66*	53.33*	86.70	51.19	16530
M-Coffee [mp]	69.05	45.71	94.47	86.11	92.45	43.85	86.79	**59.37**	90.41	56.63	90.28	57.62	15306
M-Coffee [mps]	67.98	44.58	94.40	85.75	92.28	41.98	85.78	56.77	91.17	58.31	89.57	55.75	22062
M-Coffee [mpus]	67.44	42.74	94.53	85.86	92.04	44.78	85.03	56.10	90.12	55.80	89.22	54.69	27730
SeqAn::M-Coffee [mu]	65.84	43.87	93.81	84.25	90.91	43.12	84.87	52.87	91.49	55.80	87.61	52.94	5858
SeqAn::M-Coffee [mp]	68.99	45.05	94.36	85.93	92.73	45.90	**86.84**	58.40	91.88	57.02	90.66	57.88	5400
SeqAn::M-Coffee [mps]	69.00	45.34	94.53	86.05	**93.05**	**46.78**	86.41	58.73	93.07	61.31	**90.72**	56.50	8274
SeqAn::M-Coffee [mpus]	**69.68**	**46.89**	**94.70**	**86.16**	92.55	46.56	86.53	58.90	**93.29**	**62.39**	90.06	**58.94**	12455

The total running time is reported in the last column. The number of multiple alignment files in each reference set is given in parentheses. All scores have been multiplied by 100 and the best program is shown in bold face for each column. The wilcoxon rank test was used to assess significant differences from the best method, indicated by *($P < 0.05$) or **($P < 0.01$). The submethods for all meta-methods (M-Coffee, SeqAn::M-Coffee) are given in brackets with m, MAFFT; p, ProbCons, u, MUSCLE and s, SeqAn::T-Coffee. For the meta-methods we excluded the running time of the submethods to highlight the overhead induced by the meta-method.

Table 2. PREFAB 4.0: average Q scores for all PREFAB alignments. Alignments have been subdivided according to sequence identity ranges

Aligner	≤10% (240)	10−20% (620)	20−30% (499)	30−40% (114)	40−100% (209)	CPU time (s)
AMAP	14.76**	45.92**	78.96**	90.33**	95.19*	48980
DIALIGN-T	21.25**	49.06**	77.61**	87.70**	96.30	23142
MAFFT	33.41	64.38	85.47	93.91	96.92	6078
MUSCLE	27.31**	58.03**	82.58**	91.59	95.90	4008
ProbCons	32.46	63.07*	85.54	93.22	96.24*	75272
SeqAn::T-Coffee	29.28**	61.29**	83.46**	91.84*	96.40	62455
T-Coffee	27.52**	56.62**	82.19**	89.67**	**96.98**	263434
M-Coffee [mu]	30.13*	61.85*	84.60*	93.18	96.25	79560
M-Coffee [mp]	**34.51**	64.87	86.20	93.34	96.65	77118
M-Coffee [mps]	33.26	64.73	85.75	93.44	96.75	97341
M-Coffee [mpus]	33.09	64.77	85.89	93.56	96.30	118813
SeqAn::M-Coffee [mu]	31.46	62.72	85.30*	93.59	96.22	29968
SeqAn::M-Coffee [mp]	34.43	65.50	**86.54**	93.49	96.76	28946
SeqAn::M-Coffee [mps]	33.80	65.41	**86.54**	93.61	96.60	42746
SeqAn::M-Coffee [mpus]	33.67	**65.63**	86.50	**93.92**	96.36	59559

The number of multiple alignment files in each subset is given in parentheses. All scores have been multiplied by 100 and the best program is shown in bold face for each column. The wilcoxon rank test was used to assess significant differences from the best method, indicated by * ($P < 0.05$) or ** ($P < 0.01$). The submethods for all meta-methods (M-Coffee, SeqAn::M-Coffee) are given in brackets with m, MAFFT; p, ProbCons; u, MUSCLE and s, SeqAn::T-Coffee. For the meta-methods we excluded the running time of the submethods to highlight the overhead induced by the meta-method. All methods show similar levels of score variability and variability tends to decrease for higher sequence identity ranges.

the BAliBASE scoring program to calculate the sum-of-pairs score (SP) and the total column score (TC) on the core blocks region. The results on PREFAB are shown in Table 2. We used all reference files of version 4.0 and scored the alignments with the qscore program distributed with the PREFAB database. We report the Q score that is according to the authors of the program equivalent to the BAliBASE SP score. Results obtained on protein benchmarks confirm previously reported results: MAFFT and ProbCons show the highest accuracy while MUSCLE and T-Coffee display similar albeit slightly lower levels of accuracy. M-Coffee delivers the best results when combining MAFFT and ProbCons, although this protocol does not manage to convincingly outperform the stand-alone version of MAFFT. Scores improve when the SeqAn library is used as a meta-method. SeqAn::M-Coffee outperforms most of the equivalent M-Coffee protocols, thus suggesting an improvement resulting from the segment-based extension. It is also worth mentioning that SeqAn::M-Coffee outperforms its constituting methods (including MAFFT) in most cases, thus suggesting that this method is currently

Table 3. Alignment of six adenoviruses: running time and alignment quality of an alignment of six adenoviruses. The number of columns with at least 6, 5, 4 and 3 identical characters are reported together with the average identity

Aligner	=6	≥5	≥4	≥3	Average identity	CPU time (s)
DIALIGN-T	7888	12161	18187	27690	48%	1259
SeqAn::T-Coffee	12795	18525	25147	32396	63%	1751
MAFFT*	12450	18011	24624	32084	62%	118
MUSCLE*	50	817	5257	21849	38%	673
SeqAn::T-Coffee*	**12911**	**20078**	**27011**	**33147**	**65**%	328

one of the best available multiple aligners. This improvement over M-Coffee comes along with a significant improvement in performance, with SeqAn::M-Coffee only requiring half of the CPU time of the original M-Coffee (see Table 1 and Table 2).

We then evaluated the ability of SeqAn::T-Coffee to deal with large-scale alignments. For that purpose, we ran all available packages on a set of six adenoviruses obtained from the NCBI server (Accession: NC_001460, NC_004001, NC_001405, NC_002067, NC_003266 and NC_001454). Since no gold reference is available for these long DNA sequences, we merely evaluated the ability of the programs to maximize the level of identity within the final multiple sequence alignment. Our accuracy measure was therefore the level of sequence identity in each column. We report the number of columns with at least 6, 5, 4 and 3 identical characters together with the running times of the programs and the average identity in Table 3. We first tried all the programs using the same accuracy options as for the amino acid alignments, possibly turning on some kind of DNA switch. Using this setting all programs reported an allocation error, except DIALIGN-T and SeqAn::T-Coffee. We included both tools in Table 3 and excluded all other tools reporting an allocation error. We then tried to adapt the other programs to this kind of alignment task using various command-line options. In cases where we succeeded, we included the results of the best settings in Table 3 and added a * to the methods. For SeqAn::T-Coffee, we also included a second method marked with a * that does not use local and global alignments. This method's set of input matches consists of BLAST matches and matches retrieved from pairwise comparisons using the longest common subsequence algorithm. Since most programs reported an allocation error on this set of genomes, we also analyzed a smaller set of closely related virus serotypes causing dengue fever (Accession: NC_001477, NC_001474, NC_001475 and NC_002640). All programs managed to align this set of sequences, except ProbCons and T-Coffee. Using the same notation and analysis as in Table 3 we report the results of all aligners on this set in Table 4.

Table 4. Alignment of virus serotypes: running time and alignment quality of an alignment of the four serotypes causing dengue fever. The number of columns with at least four and three identical characters are reported together with the average identity

Aligner	=4	≥3	Average identity	CPU time (s)
AMAP	930	1390	12%	179
DIALIGN-T	3635	6909	57%	46
MAFFT	5452	7948	69%	33
MUSCLE	5470	7994	69%	76
SeqAn::T-Coffee	5512	8044	69%	69
MAFFT*	5457	7949	69%	2
MUSCLE*	5481	8000	69%	19
SeqAn::T-Coffee*	**5748**	**8321**	**71**%	25
T-Coffee*	5492	8051	69%	98

4 DISCUSSION

We presented a new graph-based approach to the multiple sequence alignment problem. Our approach was especially apt to improve the meta-method M-Coffee and we showed that SeqAn::M-Coffee is more accurate and faster than the original M-Coffee. Our primary alignment method SeqAn::T-Coffee was comparable in performance to the currently best tools ProbCons and MAFFT and was consistently better than the original T-Coffee. SeqAn::T-Coffee is faster than ProbCons but due to the computation of all pairwise alignments slower than MAFFT. We plan to further improve SeqAn::T-Coffee by selecting informative pairs. We are also planning to take into account the level of segmentation induced by the procedure. Since this segmentation is a property of the data, it would probably make sense to use it further, either for estimating the alignment accuracy, or maybe as an indicator of the phylogenetic relationship between the considered sequences. Furthermore, since high segmentation levels work against the algorithm and slow it down considerably, one could identify the sequences causing the highest level of fragmentation and delay their integration until a second iteration. This could easily be implemented using the double progressive algorithm of MUMMALS (Pei and Grishin, 2006) or PROMALS (Pei and Grishin, 2007). Extending SeqAn::T-Coffee with an iterative refinement loop would probably also make sense. The usefulness of iteration is well supported by the most accurate aligners (i.e. MAFFT and MUSCLE), and a recent publication (Wheeler and Kececioglu, 2007) suggests that well designed iterations could probably improve most of the existing methods. For long DNA sequences we have illustrated SeqAn::T-Coffee's ability to compute the most accurate alignments. However, these results are only preliminary since proper benchmarks are not yet available for the validation of long nucleotide sequence alignments.

Extensions and applications of segment-based alignments are numerous. The approach lends itself to (1) the comparison of different assemblies, (2) an improved alignment of sequences with external segment information (e.g. genes or structural features), (3) an identification of conserved blocks and (4) the rapid computation of consensus sequences in assembly projects. We plan to investigate these possibilities in the near future.

Conflict of Interest: none declared.

REFERENCES

Abouelhoda,M.I. and Ohlebusch,E. (2003) Multiple genome alignment: chaining methods revisited. In *Proceedings of the 14th Annual Symposium on Combinatorial pattern matching (CPM) 2003*. pp. 1–16.

Althaus,E. *et al*. (2002). Multiple sequence alignment with arbitrary gap costs: computing an optimal solution using polyhedral combinatorics. In *Proceedings of the 1st European Conference on Computational Biology (ECCB 2002)*, pp. 4–16.

Althaus,E. *et al*. (2006). A branch-and-cut algorithm for multiple sequence alignment. *Math. Prog.*, **105**,387–425.

Altschul,S.F. *et al*. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Carrillo,H. and Lipman,D. (1988) The multiple sequence alignment problem in biology. *SIJAM: SIAM Journal on Applied Mathematics*, **48**.

Cormen,T.H. *et al*. (2001) *Introduction to Algorithms*. MIT Press, Cambridge, MA.

Delcher,A.L. *et al*. (1999) Alignment of whole genomes. *Nucl. Acids. Res.*, **27**, 2369–2376.

Chuong,B. *et al*. (2004) ProbCons: probabilistic consistency-based multiple alignment of amino acid sequences. In Deborah,L. McGuinness and George Ferguson, (eds), *AAAI*, AAAI Press / The MIT Press, pp. 703–708.

Döring,A. *et al* (2008). SeqAn - an efficient, generic C++ library for sequence analysis. *BMC Bioinformatics*, **9**,11.

Edelsbrunner,H. *Dynamic Data Structures for Orthogonal Intersection Queries. Technical Report F59*, Technical University of Graz,1980.

Edgar,R.C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, **32**, 1792–1797.

Feng,D.-F. and Doolittle,R.F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, **25**, 351–360.

Gotoh,O. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, **162**, 705–708.

Gotoh,O. (1990) Consistency of optimal sequence alignments. *BMB.*, **52**.

Halpern,A.L. *et al*. (2002) Segment match refinment and applications. In *Proceedings of the 2nd Workshop on Algorithms Bioinformatics (WABI-02)*. Springer-Verlag, London, UK, pp. 126–139.

Hirschberg,D.S. (1975) A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, **18**, 341–343.

Jacobson,G. and Vo,K.-P. (1992) Heaviest increasing/common subsequence problems. In *Proceedings of the 3rd Annual Symposium on Combinatorial Pattern Matching (CPM)*. Springer-Verlag, London, UK, pp. 52–66.

Katoh,K. *et al* (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res.*, **30**, 3059–3066.

Kececioglu,J.D. (1993) The maximum weight trace problem in multiple sequence alignment. In *CPM '93: Proceedings of 4th Annual Symposium on Combinatorial Pattern Matching*. Springer-Verlag, London, UK, pp. 106–119.

Lenhof,H.-P. *et al*. (1999) An exact solution for the segment-to-segment multiple sequence alignment problem. *Bioinformatics*, **15**, 203–210.

Lermen,M. and Reinert,K. (2000) The practical use of the A* algorithm for exact multiple sequence alignment. *J. Comp. Biol.*, **7**, 655–671.

Lipman,D.J. *et al*. (1989) A tool for multiple sequence alignment. *Proc. Natl Acad. Sci. USA*, **86**, 4412–4415.

Myers,G. and Miller,W. (1995) Chaining multiple-alignment fragments in sub-quadratic time. In *Proceedings of 6th Annual SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*. Society for Industrial and Applied Mathematics, San Francisco, California, pp. 38–47.

Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Molecular Biol.*, **48**, 443–453.

Notredame,C. *et al*. (2000) T-Coffee: a novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.*, **302**, 205–217.

Pei,J. and Grishin,N.V. (2007) MUMMALS: multiple sequence alignment improved by using hidden Markov models with local structural information. *Nucleic Acids Res.*, **34**, 4364–4374.

Pei,J. and Grishin,N.V. (2006) PROMALS: towards accurate multiple sequence alignments of distantly related proteins. *Bioinformatics*, **23**, 802–808.

Reinert,K. *et al*. (2000) An iterative methods for faster sum-of-pairs multiple sequence alignment. *Bioinformatics*, **16**, 808–814.

Saitou,N. and Nei,M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, **4**, 406–425.

Sankoff,A. and Kruskal,J. (1983) *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, MA.

Schwartz,A.S. and Pachter,L. (2007) Multiple alignment by sequence annealing. *Bioinformatics*, **23**, e24–e29.

Sokal,R.R. and Michener,C.D. (1958) A statistical method for evaluating systematic relationships. *Univ. Kansas Sci. Bull.*, **38**, 1409–1438.

Subramanian,A.R. *et al*. (2005) DIALIGN-T: an improved algorithm for segment-based multiple sequence alignment. *BMC Bioinformatics*, **6**, 66.

Szklarczyk,R. and Heringa,J. (2006) AuberGene–a sensitive genome alignment tool. *Bioinformatics*, **22**, 1431–1436.

Thompson,J.D. *et al*. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4680.

Thompson,J.D. *et al*. (2005) BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins*, **61**, 127–136.

Thompson,J.D. *et al*. (1999) BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, **15**, 87–88.

Wallace,I.M. *et al*. (2006) M-Coffee: combining multiple sequence alignment methods with T-Coffee. *Nucleic Acids Res.*, **34**, 1692–1699.

Wang,L. and Jiang,T. (1994) On the complexity of multiple sequence alignment. *J. Comput. Biol.*, **1**, 337–348.

Waterman,M.S. and Eggert.M. (1987) A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons. *J. Mol. Biol*, **197**, 723–728

Wheeler,T.J. and Kececioglu,J.D. (2007) Multiple alignment by aligning alignments. *Bioinformatics*, **23**, 559–568.