Alignment Metric Accuracy

Ariel S Schwartz Eugene W Myers Lior Pachter*

Computer Science Division
University of California Berkeley
{sariel,gene}@cs.berkeley.edu

Department of Mathematics
University of California Berkeley
lpachter@math.berkeley.edu

February 4, 2008

Abstract

We propose a metric for the space of multiple sequence alignments that can be used to compare two alignments to each other. In the case where one of the alignments is a reference alignment, the resulting accuracy measure improves upon previous approaches, and provides a balanced assessment of the fidelity of both matches and gaps. Furthermore, in the case where a reference alignment is not available, we provide empirical evidence that the distance from an alignment produced by one program to predicted alignments from other programs can be used as a control for multiple alignment experiments. In particular, we show that low accuracy alignments can be effectively identified and discarded.

We also show that in the case of pairwise sequence alignment, it is possible to find an alignment that maximizes the expected value of our accuracy measure. Unlike previous approaches based on expected accuracy alignment that tend to maximize sensitivity at the expense of specificity, our method is able to identify unalignable sequence, thereby increasing overall accuracy. In addition, the algorithm allows for control of the sensitivity/specificity tradeoff via the adjustment of a single parameter. These results are confirmed with simulation studies that show that unalignable regions can be distinguished from homologous, conserved sequences.

Finally, we propose an extension of the pairwise alignment method to multiple alignment. Our method, which we call AMAP, outperforms existing protein sequence multiple alignment programs on benchmark datasets. A webserver and software downloads are available at `http://bio.math.berkeley.edu/amap/`.

1 Introduction

A recent survey on sequence alignment [2] discusses a number of important problems and challenges that need to be overcome in order to facilitate large-scale comparative analysis of the multiple genomes currently being sequenced. Among these, the following two problems are highlighted:

1. "As suggested [9], methods to evaluate alignment accuracy. This goes at the core of the problem: which regions are alignable, and what is a correct alignment?"

2. "A definition of *alignability* – at what point is it no longer possible to do meaningful sequence alignment. Or rather, at what point can one conclude that two sequences are no longer related?"

Furthermore, the development of "rigorous methods for evaluating the accuracy of an alignment" and the need for "improved pairwise alignment with a statistical basis" are singled out as the most pressing challenges for the alignment community.

*Corresponding author

Two commonly used alignment accuracy measures are the *developer* (f_D) and the *modeler* (f_M) measures [15]. These measures correspond to evaluating the sensitivity (number of correctly matched pairs divided by the number of matched pairs in the reference alignment) and specificity (number of correctly matched pairs divided by the number of matched pairs in the predicted alignment) of matched pairs in the predicted alignment respectively. Both of these measures have a problem, in that they do not account for gap columns. [3] defined the *agreement* score, as the fraction of pairwise columns in the predicted alignment that agree with the reference alignment. While the agreement score does consider gap columns, it is not symmetric, since the number of columns in the predicted alignment can differ from the number of columns in the reference alignment.

What properties should an alignment accuracy measure satisfy? If we think of accuracy measure as a "distance", then if h^i and h^j are two alignments and we denote their distance by $d(h^i, h^j)$, we would like to have:

$$\begin{aligned} d(h^i, h^j) &\geq 0, \\ &= 0 \text{ if, and only if, } h^i \sim h^j, \quad (1) \\ d(h^i, h^j) &= d(h^j, h^i), \quad (2) \\ d(h^i, h^k) &\leq d(h^i, h^j) + d(h^j, h^k). \quad (3) \end{aligned}$$

The first condition specifies that the distance between two alignments should be non-negative and should be 0 if, and only if, the two alignments are equivalent (a useful definition of equivalence is given in section 2). The second requirement specifies that the distance should be symmetric. For example, comparing a prediction with a reference alignment should be the same as comparing the reference alignment to the prediction. The third requirement ensures a certain consistency: the distance between two predictions should be less than the sum of the distances from the predictions to a reference alignment (the triangle inequality). In other words, an accuracy measure should be based on a *metric*. Furthermore, the accuracy measure should account for unalignable sequence. For example, if two sequences are unrelated, the true alignment contains only gaps (regardless of order), and a good accuracy measure should reflect that. Note that although metrics on the space of *sequences* have been constructed [16], a metric for alignment accuracy should be defined on the space of *sequence alignments*, and should measure the distance between alignments not sequences.

In section 2 we define an alignment metric, and a new accuracy measure, called *AMA*, based on this metric. Next, we propose an algorithm which maximizes the expected AMA value given an underlying probabilistic model for mutation of sequences. We term this alignment strategy AMAP. The algorithm is explained in detail in section 3, where we show that a single parameter we term *gap-factor* (G_f) can be used to adjust the sensitivity/specificity tradeoff. A special case of AMAP, where we set $G_f = 0$, is the Maximum Expected Accuracy (MEA) alignment algorithm introduced in [6, 8] and used in ProbCons [5], and Pecan [13].

In section 4 we show how existing algorithms perform when judged using AMA, and contrast this with the developer score, which is the standard measure used in most papers [5, 7, 17]. Since the developer score is a measurement of sensitivity of aligned pairs, and algorithms have traditionally been judged by it, we find that existing algorithms are heavily biased in favor of sensitive, rather than specific alignments. An extreme case of this can be seen in Table 5 where we show that existing algorithms align large fractions of completely unrelated sequences. We also see that multiple alignments produced by different programs differ considerably from each other, even though they may appear to perform similarly when judged only by the developer score.

In a different application of the alignment metric, we show that in the typical case where reference alignments are not available for judging the success of multiple alignment experiments, the metric can be used as a control by measuring the distances between alignments predicted by different programs.

Finally, we analyze the performance of the new AMAP algorithm, using the SABmark dataset [19] and simulated datasets, and show that AMAP compares favorably to other programs.

2 Metric Based Alignment Accuracy

Following the notation in [12], an alignment of a pair of sequences $\sigma^1 = \sigma^1_1\sigma^1_2\cdots\sigma^1_n$ and $\sigma^2 = \sigma^2_1\sigma^2_2\cdots\sigma^2_m$ can be represented by an *edit string* h over the *edit alphabet* $\{H, I, D\}$, where H stands for homology, I for insertion, and D for deletion. Equivalently, if $\mathcal{A}_{n,m}$ is the set of all alignments of a sequence of length n to a sequence of length m, and $h \in \mathcal{A}_{n,m}$, then h can be represented by a path in the *alignment graph*, or a sequence of pairs of the form $(\sigma^1_i \diamond \sigma^2_j)$, $(\sigma^1 \diamond -)$, or $(\sigma^2 \diamond -)$ where the symbol $\diamond$ is used to indicate the alignment of two characters. For $h \in \mathcal{A}_{n,m}$ let

- $h_H = \{(i,j) : (\sigma^1_i \diamond \sigma^2_j) \in h\}$,
- $h_D = \{i : (\sigma^1_i \diamond -) \in h\}$,
- $h_I = \{j : (\sigma^2_j \diamond -) \in h\}$.

Less formally, h_H is the set of position pairs in σ^1 and σ^2 that are aligned according to h, h_D is the set of position in σ^1 that are gapped, and h_D is the set of position in σ^2 that are gapped. Note that for any $h \in \mathcal{A}_{n,m}$

$$|h_H| + |h_D| = n \text{ and } |h_H| + |h_I| = m. \quad (4)$$

Two alignments are equivalent if they align the same character pairs, while the order of insertions and deletions between any two consecutive aligned pairs is redundant. We therefore define:

$$h^i \sim h^j \text{ if and only if } h^i_H = h^j_H \forall h^i, h^j \in \mathcal{A}_{n,m}. \quad (5)$$

Note that $h^i_H = h^j_H$ if and only if $h^i_I = h^j_I$ and $h^i_D = h^j_D$. We can therefore use the following equivalent definition:

$$h^i \sim h^j \text{ if and only if } h^i_I = h^j_I \text{ and } h^i_D = h^j_D \quad \forall h^i, h^j \in \mathcal{A}_{n,m}. \quad (6)$$

We say that two alignments are *distinct* if they are not equivalent. The number of distinct alignments is in bijection with lattice paths in the square grid (proof omitted):

Proposition 1. *The number of distinct alignments in* $\mathcal{A}_{n,m}$ *is* $\binom{n+m}{m}$.

Given a predicted alignment h^p and a reference alignment h^r, the f_D and f_M measures are defined:

$$f(h^i, h^j) = \frac{|h^i_H \cap h^j_H|}{|h^i_H|} \quad (7)$$

$$f_D(h^p, h^r) = f(h^r, h^p) = \frac{|h^r_H \cap h^p_H|}{|h^r_H|} \quad (8)$$

$$f_M(h^p, h^r) = f(h^p, h^r) = \frac{|h^r_H \cap h^p_H|}{|h^p_H|} \quad (9)$$

Note that both measures do not explicitly use the I and D characters in h^r and h^p, and are not well defined when h^r or h^p do not include any H characters.

A distance function between any two alignments $h^i, h^j \in \mathcal{A}_{n,m}$ is needed in order to evaluate the quality of a predicted alignment given a reference alignment. Such a distance function should satisfy:

$$d(h^i, h^j) \geq 0 \qquad \forall h^i, h^j \in \mathcal{A}_{n,m}, \quad (10)$$

$$d(h^i, h^j) = 0 \text{ iff } h^i \sim h^j \qquad \forall h^i, h^j \in \mathcal{A}_{n,m}, \quad (11)$$

$$d(h^i, h^j) = d(h^j, h^i) \qquad \forall h^i, h^j \in \mathcal{A}_{n,m}, \quad (12)$$

$$d(h^i, h^j) + d(h^j, h^k) \geq d(h^i, h^k) \qquad \forall h^i, h^j, h^k \in \mathcal{A}_{n,m}. \quad (13)$$

While these requirements are not satisfied by (7), they are satisfied by the following:

$$\begin{aligned} d(h^i, h^j) &= 2|h^i_H| + |h^i_I| + |h^i_D| - 2|h^i_H \cap h^j_H| \\ &\quad -|h^i_I \cap h^j_I| - |h^i_D \cap h^j_D| \\ &= 2|h^j_H| + |h^j_I| + |h^j_D| - 2|h^i_H \cap h^j_H| \\ &\quad -|h^i_I \cap h^j_I| - |h^i_D \cap h^j_D| \\ &= n + m - 2|h^i_H \cap h^j_H| \\ &\quad -|h^i_I \cap h^j_I| - |h^i_D \cap h^j_D|). \end{aligned} \quad (14)$$

Proposition 2. $d(h^i, h^j)$ *is a finite metric for* $\mathcal{A}_{n,m}$.

Proof: It is easy to see that $d(h^i, h^j)$ satisfies requirements (10), (11), and (12). We need to show that it satisfies the triangle inequality (13). Let $U_{ij} = 2|h^i_H \cap h^j_H| + |h^i_I \cap h^j_I| + |h^i_D \cap h^j_D|$, and $U_{ijk} = 2|h^i_H \cap h^j_H \cap h^k_H| + |h^i_I \cap h^j_I \cap h^k_I| + |h^i_D \cap h^j_D \cap h^k_D|$. Using the fact that $U_{ik} - U_{ijk} \geq 0$ and $U_{ij} + U_{jk} - U_{ijk} \leq n+m$, we have that $d(h^i, h^j) + d(h^j, h^k) - d(h^i, h^k) = n + m - U_{ij} - U_{jk} + U_{ik} = n + m - (U_{ij} + U_{jk} - U_{ijk}) + U_{ik} - U_{ijk} \geq 0$.

Example 3 (Metric for $\mathcal{A}_{2,2}$). *By Proposition 1, there are six distinct alignments in $\mathcal{A}_{2,2}$. The metric is:*

	HH	*HDI*	*DIH*	*IHD*	*DHI*	*DDII*
HH	0	2	2	4	4	4
HDI	2	0	4	3	3	2
DIH	2	4	0	3	3	2
IHD	4	3	3	0	4	2
DHI	4	3	3	4	0	2
DDII	4	2	2	2	2	0

Intuitively, the distance between two alignments is the total number of characters from both sequences that are aligned differently in the two alignments. Alternatively, the quantity $g(h^i, h^j) = 1 - \frac{d(h^i,h^j)}{n+m}$ is a convenient similarity measure that can be interpreted as the fraction of characters that are aligned the same in both alignments. We therefore define the *Alignment Metric Accuracy (AMA)* of a predicted alignment h^p given a reference alignment h^r to be $g(h^p, h^r)$. The intuitive motivation for this accuracy measure is that it represents the fraction of characters in σ^1 and σ^2 that are correctly aligned, either to another character or to a gap.

AMA can easily be extended to multiple sequence alignments (MSA) by using the sum-of-pairs approach. Let $\mathcal{A}_{n_1,n_2,\ldots,n_k}$ be the space of all MSAs of k sequences of lengths n_1 to n_k. Given two MSAs $h^i, h^j \in \mathcal{A}_{n_1,n_2,\ldots,n_k}$,

$$d(h^i, h^j) = \sum_{s^1=1}^{k-1} \sum_{s^2>s^1}^{k} d(h^i_{s^1,s^2}, h^j_{s^1,s^2}), \qquad (15)$$

where $h^i_{s^1,s^2}$ is the pairwise alignment of sequences s^1, s^2 as projected from the MSA h^i with all-gap columns removed. The similarity of two MSAa is defined to be

$$g(h^p, h^r) = 1 - \frac{d(h^p, h^r)}{(k-1)\sum_{i=1}^{k} n_i}. \qquad (16)$$

Unlike standard sum-of-pairs scoring, our definition follows from the requirement that our accuracy measure should be based on a metric, and the multiple AMA retains the desirable properties of the pairwise AMA.

3 AMA Based Alignments

3.1 Maximum expected accuracy alignments

Given a probabilistic model for alignments, such as a pair-HMM, an alignment of a pair of sequences is typically obtained by the Viterbi algorithm [20], which finds the global alignment with highest probability. In the case of a pair-HMM with three states, the Viterbi algorithm is equivalent to the standard Needleman-Wunsch algorithm with affine gap scores [6]. In effect, the Viterbi algorithm maximizes the expected number of times that a predicted alignment is identical to the reference alignment ($h^p = h^r$). However, when the probability of the most likely alignment is low it might be more desirable to predict alignments that are likely to align the most number of characters correctly on average even if they are less likely to be identical to the correct alignment.

An alternative to Viterbi alignment is the *optimal accuracy* alignment [6], also called *maximum expected accuracy* (MEA) alignment [5], which maximizes the expected f_D score. The MEA alignment is calculated using a dynamic programming algorithm that finds the alignment h^p that maximizes the expected number of correctly aligned character pairs:

$$h^p = \operatorname*{argmax}_{h \in \mathcal{A}_{n,m}} \sum_{(i,j) \in h_H} \mathrm{P}(\sigma^1_i \diamond \sigma^2_j | \sigma^1, \sigma^2, \theta), \quad (17)$$

where $\mathrm{P}(\sigma^1_i \diamond \sigma^2_j | \sigma^1, \sigma^2, \theta)$ is the posterior probability that σ^1_i is homologous to σ^2_j given σ^1, σ^2 and the parameters of the model θ. In the case

of a pair-HMM, these posterior probabilities can be computed in $O(nm)$ time using the *Forward-Backward* algorithm [6].

3.2 The AMAP algorithm

While the MEA algorithm maximizes the expected f_D score it can perform very poorly on the f_M score when the reference alignment contains many unaligned characters (gaps), since it tends to over-align characters. Maximizing the expected f_M score can be done easily by only aligning the pair of characters with highest posterior probability to be homologous. This will result in an alignment with only one H character, $n-1$ D characters, and $m-1$ I characters, which in most cases will result in a poor f_D score. There is currently no alignment algorithm that allows for the adjustment of the sensitivity/specificity tradeoff (f_D / f_M tradeoff). However, we show that it is possible to maximize the expected AMA value using an algorithm similar to the original MEA algorithm. By maximizing the expected AMA, we avoid the problems of MEA alignment: in contrast to the f function, the g function is symmetric, and therefore the sensitivity ($g(h^r, h^p)$) equals the specificity ($g(h^p, h^r)$). In addition to maximizing the expected AMA value, the new algorithm, which we call AMAP allows for the modification of one parameter, which we term *gap-factor*,or G_f, to tune the f_D/f_M tradeoff.

Let $\mathrm{P}(\sigma_i^1 \diamond - | \sigma^1, \sigma^2, \theta)$ be the posterior probability that σ_i^1 is not homologous to any character in σ^2, and $\mathrm{P}(\sigma_j^2 \diamond - | \sigma^1, \sigma^2, \theta)$ the posterior probability that σ_j^2 is not homologous to any character in σ^1. AMAP should find the alignment h^p that maximizes the expected number of characters that are correctly aligned to another character or to a gap:

$$\begin{aligned} h^p = \underset{h \in \mathcal{A}_{n,m}}{\operatorname{argmax}} & \sum_{(i,j) \in h_H^p} \mathrm{P}(\sigma_i^1 \diamond \sigma_j^2 | \sigma^1, \sigma^2, \theta) + \\ & G_f \sum_{i \in h_D^p} \mathrm{P}(\sigma_i^1 \diamond - | \sigma^1, \sigma^2, \theta) + \\ & G_f \sum_{j \in h_I^p} \mathrm{P}(\sigma_j^2 \diamond - | \sigma^1, \sigma^2, \theta). \end{aligned} \quad (18)$$

Note that when $G_f = 0.5$ the algorithm maximizes the expected AMA value, while when $G_f = 0$ the expression in (18) is equal to the expression in (17), and the algorithm is identical to the original MEA algorithm, which maximized the expected f_D score. Setting G_f to higher values than 0.5 results in better f_M scores in the expense of lower f_D scores. In effect, the gap-factor parameter allows for the tuning of the f_D/f_M tradeoff.

An MEA subroutine can be used to construct multiple alignments using a number of different strategies, one of which is progressive alignment. Expected AMA maximization was performed in this context by using the ProbCons platform (the code is available in open source under the Gnu public license) with the MEA algorithm modified so that the developer score is no longer maximized. Our resulting AMAP algorithm also omits the heuristic consistency transformation of ProbCons, and simply aligns pairwise alignments along a guide tree.

4 Results

4.1 Performance of existing programs on the SABmark datasets

We began by assessing the performance of existing programs on the SABmark 1.65 [19] datasets with the goal of comparing alignment metric accuracy with previously used measures. SABmark includes two sets of pairwise reference alignments with known structure from the ASTRAL [4] database. The Twilight Zone set contains 1740 sequences with less than 25% identity divided into 209 groups based on SCOP folds [10]. The Superfamilies set contains 3280 sequences with less than 50% identity divided into 425 groups. Additionally, each dataset has a "false positives" version, which contains unrelated sequences with the same degree of sequence similarity in addition to the related sequences.

Table 1 shows the performance of a number of existing alignment programs as measured by the developer, modeler, and AMA accuracy measures on the four SABmark 1.65 datasets. Methods tested include Align-m 2.3 [19], CLUSTALW

	Twilight			Superfamilies			Twilight-FP			Superfamilies-FP		
Program	f_D	f_M	AMA	f_D	f_M	AMA	f_D	f_M	AMA	f_D	f_M	AMA
Align-m	21.6	**23.6**	**51.7**	49.2	**45.6**	**56.9**	17.8	**6.4**	**81.5**	44.8	**16.8**	**77.5**
CLUSTALW	25.6	14.7	24.9	54.0	38.1	43.8	20.4	2.4	35.5	50.9	7.4	37.0
MUSCLE	27.3	16.4	27.6	56.3	40.3	46.4	19.4	2.3	37.1	49.7	7.5	38.9
ProbCons	**32.1**	21.1	37.3	**59.8**	44.4	51.8	**26.7**	4.4	55.7	56.0	10.9	55.0
T-Coffee	29.4	19.6	35.6	58.4	43.7	50.9	26.5	4.2	54.1	**57.0**	11.0	54.4

Table 1: **Performance of aligners on the SABmark benchmark datasets.** Entries show the average developer (f_D), modeler (f_M) and alignment metric accuracy (AMA). Best results are shown in bold. All numbers have been multiplied by 100.

1.83 [18], MUSCLE 3.52 [7], ProbCons 1.1 [5] and T-Coffee 2.49 [11]. The results highlight the inherent sensitivity/specificity tradeoff. While ProbCons and T-Coffee have the best developer scores, Align-m has the best modeler scores. It is not clear which program outperforms the others. Programs with higher sensitivity tend to over-align unalignable regions, which results in lower specificity. We would like to answer the question, which program produces alignments that are the closest to the reference alignments? This is exactly the interpretation of the new AMA measure. Using this measure it is clear that Align-m is the most accurate alignment program among the ones tested on the SABmark benchmark datasets.[1]

4.2 Controls for multiple alignment experiments

The reference alignments in SABmark are themselves somewhat subjective, as they are based on structural alignment programs. A recurring question has been how to judge the accuracy of alignment in the absence of a reference. To demonstrate how AMA is useful for that, we compared the total distance and average similarity (g) between the alignments produced by the five alignment programs, two variants of the AMAP algorithm, and the reference alignments. Table 2 shows these values averaged over the entire SABmark dataset. An interesting observation is that there is a correlation between the distances of the alignments of any given program to the reference alignment, and their distances to the closest alignments of other programs. For example, CLUSTALW alignments are 37% similar on average to the reference SABmark alignments, and 39.5% similar to AMAP alignments, while Align-m alignments are 67% similar to the reference alignments, and 68.7% similar to AMAP-4 alignments.

We propose to use the similarity of predicted alignments from different alignment program as a control before using a predicted alignment. Figure 1 shows the correlation between the accuracy (AMA) of of CLUSTALW alignments of the Twilight-FP and Superfamilies-FP datasets, and their maximum similarity to any of the alignments produced by the other four programs. It is evident that there is a strong correlation between the two values. We observed similar correlation for the other alignment programs (see supplementary data). In the case of the two datasets with no false positives (Twilight and Superfamilies) the maximum similarity behaves more as an upper bound rather than a strong predictor for the AMA value (data not shown). This is still useful for discarding alignments that are not similar to other predicted alignments, since they are very unlikely to be similar to the true alignment.

To summarize, the following protocol can be used to identify bad alignments in the absence of a reference alignment:

1. Align the target sequences with a preferred alignment program.

2. Align the target sequences with all other

[1]Note that the SABmark benchmark dataset was compiled by the same authors as Align-m.

	Align-m	CLUSTALW	MUSCLE	ProbCons	T-Coffee	AMAP	AMAP-4	Reference
Align-m		37.3	39.9	52.8	50.4	64.7	68.7	**67.0**
CLUSTALW	45.0		38.2	38.1	39.1	39.5	35.9	**37.0**
MUSCLE	43.8	49.4		43.2	42.3	43.9	38.9	**39.2**
ProbCons	32.0	48.7	47.1		52.0	67.2	53.4	**51.1**
T-Coffee	30.1	47.0	45.9	38.7		56.6	50.9	**50.1**
AMAP	15.5	45.1	43.7	29.8	29.4		72.9	**64.4**
AMAP-4	13.3	45.6	44.3	32.1	29.9	11.6		**70.2**
Reference	**13.7**	**44.1**	**43.0**	**31.1**	**28.6**	**13.8**	**11.4**	

Table 2: **Total distance and average similarity (g) of different aligners on the SABmark dataset.** Values below the diagonal show the total distance between alignments produced by different alignment programs. Values above the diagonal show the average similarity (g) between the different alignments. Distance values have been divided by one million, and similarity values have been multiplied by 100. AMAP-4 is the AMAP algorithm with gap-factor of 4.

available alignment programs.

3. Measure the similarity of the first alignment to every other alignment,

4. If the similarity of the closest alignment to the first alignment is below a certain threshold, discard the alignment.

The above procedure has no mathematical guarantees, but our empirical results show that most of the discarded alignments will have an AMA value less than the selected threshold.

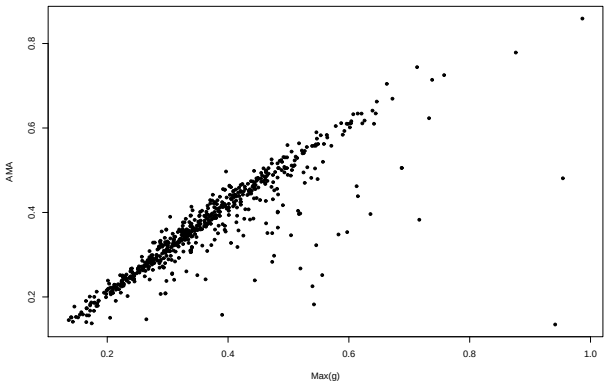

Figure 1: **Correlation between the AMA of CLUSTALW (as judged by reference alignments in SABmark), and distance to the nearest alignment produced by another program.** Each dot in the plot corresponds to one CLUSTALW alignment in the SABmark Twilight-FP and Superfamilies-FP datasets. The x coordinate represents the similarity (g) of the CLUSTALW alignment to the closest alignment produced by one of four other programs (Align-m, MUSCLE, ProbCons, T-Coffee). The y coordinate represents the Alignment Metric Accuracy (AMA) of the CLUSTALW alignment as judged by the reference SABmark alignment.

4.3 Performance of the AMAP algorithm

Next, we investigated, using pairwise alignments, whether the AMAP algorithm can improve on the Viterbi and the Maximum Expected Accuracy (MEA) alignment algorithms for maximizing the AMA.

We first evaluated the algorithms with the same default parameters used in ProbCons ($\delta = 0.01993$, $\varepsilon = 0.79433$, $\pi_{match} = 0.60803$, and emission probabilities based on the BLOSUM62 matrix). Table 3 shows the results of the Viterbi algorithm and the AMAP algorithm with different gap-factor values on the SABmark Twilight Zone set, and table 4 show the results on the SABmark Superfamilies set.

The results on both sets show the expected correlation between the gap-factor value and the f_M score, and the negative correlation between the gap-factor value and the f_D score. This

Default transition probabilities				"Correct" transition probabilities			
Algorithm	f_D	f_M	AMA	Algorithm	f_D	f_M	AMA
Viterbi	27.2	16.3	28.0	Viterbi	18.8	17.0	46.7
$G_f = 0$	**29.6**	17.7	29.0	$G_f = 0$	**25.9**	17.5	37.2
$G_f = 0.5$	28.1	19.7	37.4	$G_f = 0.5$	17.2	34.7	56.3
$G_f = 1$	25.8	22.8	45.2	$G_f = 1$	14.1	42.2	**57.3**
$G_f = 2$	22.4	27.6	51.5	$G_f = 2$	11.3	52.2	57.3
$G_f = 4$	18.9	33.1	54.8	$G_f = 4$	8.9	59.3	56.7
$G_f = 8$	15.9	38.9	56.4	$G_f = 8$	7.0	68.7	56.0
$G_f = 12$	14.3	43.3	56.7	$G_f = 12$	6.1	74.5	55.6
$G_f = 16$	13.1	46.1	56.8	$G_f = 16$	5.5	77.3	55.4
$G_f = 20$	12.4	48.0	**56.8**	$G_f = 20$	5.1	80.2	55.2
$G_f = 28$	11.3	**51.5**	56.7	$G_f = 28$	4.6	**83.1**	55.0

Table 3: **Performance of algorithm variants on the SABmark Twilight Zone set.** Entries show the f_D, f_M, and AMA scores of the Viterbi, and AMAP alignments with different gap-factor (G_f) values on the SABmark Twilight Zone set, which includes 209 alignment groups. The first five columns show the results using default transition probabilities, and the last five columns show the results using transition probabilities calculated for each group from the reference alignments. All scores have been averaged over groups and multiplied by 100.

Default transition probabilities				"Correct" transition probabilities			
Algorithm	f_D	f_M	AMA	Algorithm	f_D	f_M	AMA
Viterbi	53.1	38.1	44.2	Viterbi	46.8	41.3	53.3
$G_f = 0$	**54.8**	39.3	45.2	$G_f = 0$	**52.4**	40.1	49.2
$G_f = 0.5$	53.6	42.0	49.8	$G_f = 0.5$	45.4	56.1	60.5
$G_f = 1$	51.6	46.2	54.5	$G_f = 1$	41.8	63.4	**61.5**
$G_f = 2$	48.1	52.1	58.2	$G_f = 2$	37.9	70.9	61.2
$G_f = 4$	44.1	58.5	59.9	$G_f = 4$	34.1	75.9	60.0
$G_f = 6$	41.8	62.0	**60.2**	$G_f = 6$	31.9	78.4	59.2
$G_f = 8$	40.2	**64.3**	60.1	$G_f = 8$	30.5	**79.9**	58.6

Table 4: **Performance of algorithm variants on the SABmark Superfamilies set.** Entries show the f_D, f_M, and AMA scores of the Viterbi, and AMAP alignments with different gap-factor (G_f) values on the SABmark Superfamilies set, which includes 425 alignment groups. The first five columns show the results using default transition probabilities, and the last five columns show the results using transition probabilities calculated for each group from the reference alignments. All scores have been averaged over groups and multiplied by 100.

validates the prediction that the gap-factor can be used as a tuning parameter for the sensitivity/specificity tradeoff of matched characters.

When the gap-factor is set to 0.5 or higher the alignment accuracy is significantly better than Viterbi alignments, when the original MEA algorithm (AMAP with gap-factor set to 0) is used, the alignment accuracy is almost identical to that of the Viterbi algorithm. The most accurate alignments were achieved by setting the gap-factor to values higher than 0.5 (20 in the Twilight Zone set and 6 in the Superfamilies set). We suspected that this is due to the fact that the default pair-HMM parameters underes-

Parameter Settings			f_D				f_M				AMA			
e_{match}	δ	ε	0	0.5	1	Vit	0	0.5	1	Vit	0	0.5	1	Vit
30	10.0	90	**7.1**	1.1	0.8	0.6	4.0	61.7	**73.7**	2.2	9.5	**51.0**	50.9	44.7
50	10.0	90	**23.6**	3.9	2.1	12.5	15.7	77.0	**83.5**	10.8	27.2	**52.2**	51.4	30.3
60	10.0	90	**33.6**	17.7	12.2	24.6	23.2	69.5	**82.9**	23.8	34.5	**57.2**	55.9	41.2
80	10.0	90	**61.2**	53.4	46.6	53.6	45.9	72.1	**81.9**	51.7	57.3	**70.8**	70.7	62.4
80	5.0	90	**83.2**	80.8	77.5	81.1	76.3	85.3	**89.9**	77.7	78.8	**82.4**	82.2	78.6
80	2.0	98	**94.4**	93.5	92.7	92.9	89.0	95.0	**96.2**	93.6	93.0	**95.4**	95.3	94.6
80	0.9	98	**97.4**	97.2	96.6	96.8	95.2	97.5	**98.2**	96.6	96.2	**97.2**	**97.2**	96.7
30	0.1	98	**94.5**	**94.5**	93.8	90.1	93.7	93.8	**94.1**	89.3	**93.1**	**93.1**	92.6	88.5
50	0.1	98	**99.4**	**99.4**	**99.4**	99.3	**99.4**	**99.4**	**99.4**	99.3	**99.2**	**99.2**	**99.2**	99.0
unrelated			100	100	100	100	0.0	0.0	0.0	0.0	72.2	96.8	**98.4**	94.3

Table 5: **Performance of algorithm variants on simulated data.** Entries show the performance of the Viterbi algorithm (Vit), and the AMAP algorithm with different settings of the gap-factor parameter (0, 0.5, and 1) using three accuracy measures (f_D, f_M, and AMA). The first three columns show the configuration of the pair-HMM parameters e_{match} (match emission probability), δ (gap initiation probability) and ε (gap extension probability), except for the last row for which random unrelated sequences have been aligned. Best results for every parameter configuration and measure are shown in bold. All numbers have been multiplied by 100.

timate the probability of insertion and deletions. To validate that we calculated the "true" transition probabilities for each alignment group using the reference alignments, and repeated the experiment.

The performance of the algorithms using the "correct" transition probabilities are shown in the right columns of tables 4 and 3. As expected, with the correct parameters, the accuracy of the alignments achieved when the gap-factor is set to 0.5 are very close to the best ($G_f = 1$). Note that we did not modify the emission probabilities, which might be the reason $G_f = 0.5$ did not maximize the actual accuracy. These results show that the AMAP algorithm significantly outperforms the Viterbi and MEA algorithms (61.5 AMA compared to 53.3 and 49.2 AMA on the Superfamilies dataset, and 57.3 AMA compared to 46.7 and 37.2 on the Twilight Zone dataset). Moreover, with the adjusted parameters, the Viterbi algorithm outperforms the MEA algorithm ($G_f = 0$) on both datasets. This is due to the over-alignment problem of the MEA algorithm, which uses the expected f_D score as its objective function at the expense of the f_M and AMA scores. Note that the best AMA scores achieved with the default transition probabilities are very close to those of the correct probabilities, demonstrating that adjustment of the gap-factor parameter is able to compensate for bad estimation of the parameters of the underlying probabilistic model.

In order to further analyze the performance of the AMAP algorithm compared to the Viterbi and MEA algorithms, we also conducted simulation studies. Table 5 compares the performance of the Viterbi, MEA, and AMAP variants on different sets of simulated pairs of related and unrelated DNA sequences.

Data was simulated using a pair-HMM to generate aligned pairs of nucleotide sequences. The pair-HMM parameters included the transition probabilities δ (gap initiation) and ε (gap extension). For simplicity we fixed the initial probability π_{match} of starting in a Match state to be $1 - 2\delta$. For the emission probabilities we used a simple model that assigns equal probability ($\frac{1}{4}$) to any nucleotide in the Insert or Delete states, $\frac{e_{match}}{4}$ probability for a match in the Match state, and $\frac{1-e_{match}}{12}$ probability for a mismatch in the Match state, where e_{match} is the probability to

Program	Twilight	Superfamilies	Twilight-FP	Superfamilies-FP	Overall
Align-m	51.7	56.9	81.5	77.5	67.0
ProbCons	37.3	51.8	55.7	55.0	51.1
AMAP	46.1	56.3	75.8	75.8	64.4
AMAP-4	**52.2**	**57.9**	**84.2**	**84.6**	**70.2**

Table 6: **Performance of selected programs on the SABmark benchmark datasets.** Entries show the AMA score for each program and data set. All numbers have been multiplied by 100.

emit a pair of identical characters in the Match state.

For every setting of the parameters we generated 10 reference alignments with $\min(n, m) = 1000$. An identical pair-HMM with the same parameters was then used to compare the performance of the Viterbi algorithm and MEA algorithm with gap-factor values of (0, 0.5, and 1). We treat every set of 10 alignments as one big alignment, and calculate the accuracy ($g(h^p, h^r)$) of the predicted alignments, the f_D, and f_M scores. In addition to the simulated reference alignment generated from the pair-HMM, we also generated 10 pairs of unrelated random sequences of length 1000 each with equal probability for every character. All algorithms have been evaluated on the resulting reference alignments, which include no H characters, using the probabilities 0.8, 0.1, and 0.9 for the e_{match}, δ, and ε parameters respectively.

The simulation results demonstrate that the AMAP algorithm produces alignments that are more accurate than the Viterbi and MEA alignment algorithms on both closely related and distant sequences. As expected the best f_D scores are achieved using the MEA algorithm ($G_f = 0$), the best f_M scores when $G_f = 1$, and the best AMA when $G_f = 0.5$.

It is interesting to note that for distant sequences with larger gap initiation probability (δ), the Viterbi algorithm has better AMA score than the MEA algorithm ($G_f = 0$). This again emphasizes the main weakness of MEA, which tends to over-align unalignable regions. This problem is even more pronounced when aligning unrelated sequences. The MEA algorithm performs poorly compared to the AMAP algorithm and even the Viterbi algorithm, achieving a mere 72.2 AMA scored compared to 96.8 and 94.8 respectively. This is due to the fact that the MEA algorithm wrongly aligns 2781 character pairs, compared to 157, 316, and 572 in the case of $G_f = 1$, $G_f = 0.5$, and Viterbi alignment respectively (data not shown).

Finally, we compared the performance of the multiple sequence alignments version of the AMAP algorithm compared to ProbCons and Align-m. Table 6 shows the AMA scores of each program on the four SABmark datasets. AMAP and Align-m are clearly superior to ProbCons. While AMAP with default parameters achieves slightly lower AMA scores than Align-m, setting the gap-factor to 4 produces the most accurate alignments. This demonstrate the power of the AMAP algorithm, which can easily be tuned using a single parameter to improve alignment accuracy even when the parameters of the underlying statistical model (transition and emission probabilities of the Pair-HMM) do not fit the data very well.

5 Discussion

We have proposed a metric for the set of alignments, and shown how it can be used both to judge the accuracy of alignments, and as the basis for an optimization criteria for alignment. The importance of the metric lies in the fact that if two alignments are far from each other, we can conclude that at least one of them is inaccurate. This is a direct consequence of the triangle inequality. More importantly, we show that when alignments made by widely used software programs are compared to each other they

are far apart, thus quantitatively confirming that multiple alignment is a difficult problem. Although we see that the sensitivity of many programs is high, i.e., many of the residues that should be aligned together are correctly aligned, it is also the case that many residues are incorrectly aligned. This is particularly evident in results from the Twilight-FP and Superfamilies-FP datasets that contain unrelated sequences. If functional inferences are to be made from sequence alignments, it is therefore important to control for specificity, and not only sensitivity. Our alignment algorithm, AMAP, which maximizes the expected AMA, outperforms existing programs on benchmark datasets.

Most exiting multiple alignment benchmark datasets include only alignments of "core blocks", and it is therefore only possible to measure the sensitivity of matches (f_D), and not their specificity (f_M) or the AMA. However, the fact that it is harder to construct datasets that allow for measuring the latter two does not mean that alignment algorithms should maximize sensitivity at the expense of specificity. Our AMAP algorithm is the first to allows the user to tune the inherent sensitivity/specificity tradeoff using the gap-factor parameter. In many cases, such as when using MSA for phylogenetic tree reconstruction, or for identification of remote homology, higher specificity is preferred over higher sensitivity. In addition, as we have shown, tuning the gap-factor parameter can in some cases compensate for poor parameter estimation of the underlying probabilistic model (pair-HMM). Further work is needed to develop methods for automatic adjustment of this parameter for a given dataset when the probabilistic parameters do not fit the data very well, and a reference alignment is not available.

In the typical case where reference alignments are not available, our empirical observation that the distances between alignments correlate strongly with the accuracy of the programs that generated them can be used to discard inaccurate alignments. It is possible that more sophisticated strategies based on this principle could further help in quantitatively assessing alignment reliability.

We have not discussed the relevance of our results to DNA multiple alignments, however many of our ideas can be easily adapted. As with protein multiple alignment, the focus in DNA alignment has been on sensitivity rather than specificity. For example, whole genome alignments are often judged by exon coverage. We have focused on protein sequence in this initial study, because in certain cases reference alignments can be constructed based on structural alignments.

Finally, we mention that it is possible to formulate MEA multiple alignment using our AMA as an integer linear program using ideas similar to those in [1, 14]. In particular, it is possible to set up a program with a polynomial number of variables and constraints. It should be interesting to study the possibility of applying approximation methods to solving such programs.

Acknowledgments

A.S was partially supported by NSF grant EF 03-31494. G.M. was supported by the Max-Planck / Alexander von Humboldt International Research Prize. L.P. was partially supported by a Sloan Research Fellowship.

References

[1] Ernst Althaus, Alberto Caprara, Hans Peter Lenhof, and Knut Reinert. Multiple sequence alignment with arbitrary gap costs: Computing an optimal solution using polyhedral combinatorics. *Bioinformatics*, 18(90002):4S–16, 2002.

[2] S Batzoglou. The many faces of sequence alignment. *Briefings in Bioinformatics*, 6:6–22, 2005.

[3] M Blanchette, W J Kent, C Riemer, L Elnitski, A F A Smit, K M Roskin, R Baertsch, K Rosenbloom, H Clawson, E D Green, D Haussler, and W Miller. Aligning multiple genome sequences with the threaded blockset aligner. *Genome Research*, 14:708–715, 2004.

[4] Steven E. Brenner, Patrice Koehl, and Michael Levitt. The ASTRAL compendium for protein structure and sequence analysis. *Nucl. Acids Res.*, 28(1):254–256, 2000.

[5] Chuong B. Do, Mahathi S.P. Mahabhashyam, Michael Brudno, and Serafim Batzoglou. ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Res.*, 15(2):330–340, 2005.

[6] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis. Probablistic models of proteins and nucleic acids*. Cambridge University Press, 1998.

[7] Robert C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucl. Acids Res.*, 32(5):1792–1797, 2004.

[8] I. Holmes and R. Durbin. Dynamic programming alignment accuracy. *J. Comp. Biol.*, 5(3):493–504, 1998.

[9] W Miller. comparison of genomic sequences: Solved and unsolved problems. *Bioinformatics*, 17:391–397, 2000.

[10] A.G. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536, 1995.

[11] C Notredame, D Higgins, and J Heringa. T-coffee: A novel method for multiple sequence alignments. *Journal of Molecular Biology*, 302:205–217, 2000.

[12] L. Pachter and B. Sturmfels, editors. *Algebraic Statistics for Computational Biology*. Cambridge University Press, 2005.

[13] B. Paten. http://www.ebi.ac.uk/∼bjp/pecan/, 2005.

[14] S. Prestwich, D. Higgins, and O. O'Sullivan. Pseudo-boolean multiple sequence alignment. In *Tenth Workshop on Automated Reasoning*, 2003.

[15] J.M. Sauder, J.W. Arthur, and R.L. Dunbrack. Large-scale comparison of protein sequence alignment algorithms with structure alignments. *Proteins*, 40(1):6–22, 2000.

[16] Peter A. Spiro and Natasa Macura. A local alignment metric for accelerating biosequence database search. *Journal of Computational Biology*, 11(1):61–82, 2004.

[17] Sing-Hoi Sze, Yue Lu, and Qingwu Yang. A polynomial time solvable formulation of multiple sequence alignment. *Lecture Notes in Computer Science*, 3500:204–216, April 2005.

[18] JD Thompson, DG Higgins, and TJ Gibseon. Clustalw: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

[19] Ivo Van Walle, Ignace Lasters, and Lode Wyns. SABmark–a benchmark for sequence alignment that covers the entire known fold space. *Bioinformatics*, 21(7):1267–1268, 2005.

[20] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13(2):260–269, April 1967.