Ensemble Multiple Sequence Alignment of Proteins (DRAFT)

Dan DeBlasio[1†*], John Kececioglu[2]

1 Computational Biology Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA
2 Department of Computer Science, The University of Arizona, Tucson, AZ 85721, USA
† Work originally performed while at The University of Arizona Department of Computer Science.

* deblasio@cmu.edu

Abstract

The multiple sequence alignments computed by an aligner for different *settings* of its parameters, as well as the alignments computed by different *aligners* using their default settings, can differ markedly in accuracy. *Parameter advising* is the task of choosing a parameter setting for an aligner to maximize the accuracy of the resulting alignment. We extend parameter advising to *aligner advising*, which in contrast chooses among a set of aligners to maximize accuracy. In the context of aligner advising, *default* advising selects from a set of aligners that are using their default settings, while *general* advising selects both the aligner and its parameter setting.

In this paper, we apply aligner advising for the first time, to create a true *ensemble aligner*. Through cross-validation experiments on benchmark protein sequence alignments, we show that parameter advising boosts an aligner's accuracy beyond its default setting for virtually all of the standard aligners currently used in practice. Furthermore, aligner advising with a collection of aligners further improves upon parameter advising with any single aligner, though surprisingly the performance of default advising on testing data is actually superior to general advising due to less overfitting to training data.

The new ensemble aligner that results from aligner advising is significantly more accurate than the best single default aligner, especially on hard-to-align sequences. This successfully demonstrates how to construct out of a collection of individual aligners, a more accurate ensemble aligner.

keywords: Multiple sequence alignment, parameter advising, aligner advising, accuracy estimation, ensemble methods.

1 Introduction

While it has long been known that the multiple sequence alignment computed by an aligner strongly depends on the settings for its tunable parameters, and that different aligners using their default settings can output markedly different alignments of the same input sequences, there has been relatively little work on how to automatically choose the best parameter settings for an aligner, or the best aligner to invoke, to obtain the most accurate alignment of a given set of input sequences.

Automatically choosing the best parameter setting for an aligner on a given input was termed by Wheeler and Kececioglu [1], *parameter advising*. In their framework, an

```
d1flma    8   ... fevlknegvvAIATQgedgphlvntwnsylkv-ldgnrivvpvggmhkteanva-rde ...  63
d1ci0a    18  ... tkw-fn--------eakedpret-------------lpeaiTFSS-------Aelpsg ...  46
d1nrga    16  ... aaw-fe--------eavqcpdig-------------eanamCLAT-------Ct-rdg ...  43
d1ejea    22  ... hriltprptvMVTTVdeegninaapfsftmpvsidppvvafasapdhhtarnie-sth ...  78
d1i0ra    8   ... ykisyglyIVTSEsngrkcgqiant---vfqltskpvqiavclnkendthnavk-esg ...  61
```

(a) Lower-accuracy alignment computed by MUMMALS

```
d1flma    1   ... ------mlpgtffevlkne-----gvvAIATQg-edgph--lvntwnsylk---vldg ...  41
d1ci0a    12  ... d-dpidlftkwfneakedpretlpeaiTFSSAelpsgr----vssrillfk---eldh ...  59
d1nrga    9   ... sldpvkqfaawfeeavqcpdigeanamCLATCt-rdgk----psarmlllk---gfgk ...  56
d1ejea    11  ... s-mdfedfpvesahriltpr----ptvMVTTVd-eegn----inaapfsftmpvsidp ...  56
d1i0ra    1   ... --mdveafykisy------------glyIVTSE-sngrkcgqiantvfqlt---s-kp ...  39
```

(b) Higher-accuracy alignment computed by Opal

Fig 1. *Aligner choice affects the accuracy of computed alignments.* (a) Part of an alignment of benchmark sup_125 from the SABRE [2] suite computed by MUMMALS [3] using its default parameter choice; this alignment has accuracy value 28.9%, and Facet [4] estimator value 0.540. (b) Alignment of the same benchmark by Opal [1] using its default parameter choice, which has 49.9% accuracy, and higher Facet value 0.578. In both alignments, the positions that correspond to *core blocks* of the reference alignment, which should be aligned in a correct alignment, are highlighted in bold.

advisor takes a *set* of parameter settings, together with an *estimator* that estimates the accuracy of a computed alignment, and invokes the aligner on each setting, evaluates the accuracy estimator on each resulting alignment, and chooses the setting that gives the alignment of highest estimated accuracy. Analogously, we call automatically choosing the best aligner for a given input, *aligner advising*.

To make this concrete, Figure 1 shows an example of advising on a benchmark set of protein sequences for which a correct reference alignment is known, and hence for which the true accuracy of a computed alignment can be determined. In this example, the Facet estimator of DeBlasio and Kececioglu [5] is used to estimate the accuracy of two alignments computed by the Opal [6] and MUMMALS [3] aligners. For these two alignments, the one of higher Facet value also has higher true accuracy as well, so an advisor armed with the Facet estimator would in fact output the more accurate alignment to a user.

For a collection of aligners, this kind of advising is akin to an *ensemble* approach to alignment, which selects a solution from those output by different methods to obtain in effect a new method that ideally is better than any individual method. Ensemble methods have been studied in machine learning [7], which combine the results of different classifiers to produce a single output classification. Typically such ensemble methods from machine learning select a result by *voting*. In contrast, an advisor combines the results of aligners by selecting one via an *estimator*.

In this paper, we extend the framework of parameter advising to aligner advising, and obtain by this natural approach a true *ensemble aligner*. Moreover as our experimental results show, the resulting ensemble aligner is significantly more accurate than any individual aligner.

Related work

We very briefly summarize prior work on alignment advising. Wheeler and Kececioglu [1] introduced the notion of *parameter advisors*; Kececioglu and DeBlasio [4] investigated the construction of alignment *accuracy estimators*, resulting in the Facet estimator [5]; and DeBlasio and Kececioglu [8] investigated how to best form the set of parameter choices for an advisor, called an *advisor set*, developing an efficient approximation algorithm for finding a near-optimal advisor set for a given estimator. All this prior work applied parameter advising to boosting the accuracy of the `Opal` aligner [6]. In contrast, this paper applies *parameter advising* to all commonly-used aligners, and *aligner advising* to combine them into a new, more accurate, ensemble aligner.

To our knowledge, the only prior work on combining aligners is by Wallace, O'Sullivan, Higgins, and Notredame [9] on the `M-Coffee` aligner, Collingridge and Kelly who developed the `MergeAlign` [10] tool and by Muller, Creevey, Thompson, Arendt, and Bork [11] on the `AQUA` tool. `AQUA` chooses between an alignment computed by `MUSCLE` [12] or `MAFFT` [13] based on their `NorMD` [14] score. Our prior work [4] shows that for choosing the more accurate alignment, the `NorMD` score used by `AQUA` is much weaker than the `Facet` estimator used here for aligner advising. `M-Coffee` uses a standard progressive alignment heuristic to compute an alignment under position-dependent substitution scores whose values are determined by alignments from different aligners. `MergeAlign` produces a consensus alignment of several alignments of the same input sequences but constructing a partial order alignment graph that represents all of the alignments, the path with the highest support is then returned. As Section 3.3 later shows, when run on the same set of aligners, `M-Coffee` and `MergeAlign` are both strongly dominated by the ensemble approach of this paper.

Plan of the paper

In the next section, we review our approach to learning an alignment advisor. An advisor selects aligners and parameter values from a small set of choices that is drawn from a larger universe of all possible choices. Section 2.2 describes how we construct this universe of aligners and their parameter choices for advisor learning. Section 3 then experimentally evaluates our approach to ensemble alignment on real biological benchmarks.

2 Methods

2.1 Learning an alignment advisor

To make the paper self-contained, we briefly review our prior work on how to learn an alignment advisor. We first review the concept of *parameter advising*, which requires an estimator of alignment accuracy and a set of parameter choices for the advisor, and then summarize our prior techniques for learning both an estimator and an advisor set. In Section 3, we apply these techniques for the first time to *aligner advising* to yield a new ensemble aligner.

2.1.1 Parameter advising

The goal of parameter advising is to find the parameter setting for an aligner that yields the most accurate alignment of a given set of input sequences. The accuracy of a computed alignment is measured with respect to the "correct" alignment of the sequences (which often is not known). For special benchmark sets of protein sequences, the gold-standard alignment of the proteins, called their *reference alignment*, is usually obtained through structural alignment by finding the best superposition of the known three-dimensional structures of the proteins. Columns of the reference alignment that contain a residue from every protein in the set (where a *residue* is the amino acid at a particular position in a protein), and for which the residues in the column are all mutually close in space in the superposition of the structures, are called *core columns*.

Runs of consecutive core columns are called *core blocks*, which represent the regions of the structural alignment with the highest confidence of being correct. Given such a reference alignment with identified core blocks, the *accuracy* of a different, computed alignment is the fraction of the pairs of residues aligned in the core blocks of the reference alignment that are also aligned in the computed alignment. (So a computed alignment of 100% accuracy completely agrees with the reference on its core blocks, though it may disagree elsewhere.) The best computed alignment is one of highest accuracy, and the task of a parameter advisor is to find a setting of the tunable parameters of an aligner that yields an accurate output alignment. This setting can be highly input dependent, as the best choice of parameter values for an aligner can vary for different sets of input sequences.

When aligning sequences in practice, a reference alignment is almost never known, in which case the true accuracy of a computed alignment cannot be measured. Instead our parameter advisors rely on an accuracy *estimator* E that for an alignment A, gives a value $E(A)$ in the range $[0,1]$ that estimates the true accuracy of alignment A. An estimator should be efficiently computable and positively correlated with true accuracy.

To choose a parameter setting, an advisor takes a set of choices P, where each *parameter choice* $p \in P$ is a vector that assigns values to all the tunable parameters of an aligner, and picks the choice that yields a computed alignment of highest estimated accuracy.

Formally, given an accuracy estimator E and a set P of parameter choices, a *parameter advisor* tries each parameter choice $p \in P$, invokes an aligner to compute an alignment A_p using choice p, and then selects the parameter choice p^* that has maximum estimated accuracy:

$$p^* \;\in\; \operatorname*{argmax}_{p \,\in\, P} \Big\{ E(A_p) \Big\}.$$

Since the advisor runs the aligner $|P|$ times on a given set of input sequences, a crucial aspect of parameter advising is finding a small set P for which the true accuracy of the output alignment A_{p^*} is high.

To construct a good advisor, we need to find a good estimator E and a good set P. The estimator and advisor set are learned on training data consisting of benchmark sets of protein sequences for which a reference alignment is known. The learning procedure tries to find an estimator E and set P that maximize the true accuracy of the resulting advisor on this training data, which we subsequently assess on separate testing data.

Note that the process of advising is fast: for a set P of k parameter choices, advising involves computing k alignments under these choices, which can be done in parallel, evaluating the estimator on these k alignments, and taking a max. (Section 3.8 gives actual running times.) The separate process of training an advisor, by learning an estimator and an advisor set as we review next, is done once, off-line, before any advising takes place.

2.1.2 Learning an accuracy estimator

Kececioglu and DeBlasio [4, 16] present an efficient approach for learning an accuracy estimator that is a linear combination of real-valued alignment feature functions, based on solving a large-scale linear programming problem. This approach resulted in the `Facet` estimator [5], which is currently the most accurate estimator for parameter advising [4, 8].

This approach assumes we have a collection of d real-valued feature functions $g_1(A), \ldots, g_d(A)$ on alignments A, where these functions g_i are positively correlated with true accuracy. The alignment accuracy estimator E is a linear combination of these functions, $E(A) = \sum_{1 \le i \le d} c_i \, g_i(A)$, where the coefficents c_i specify the

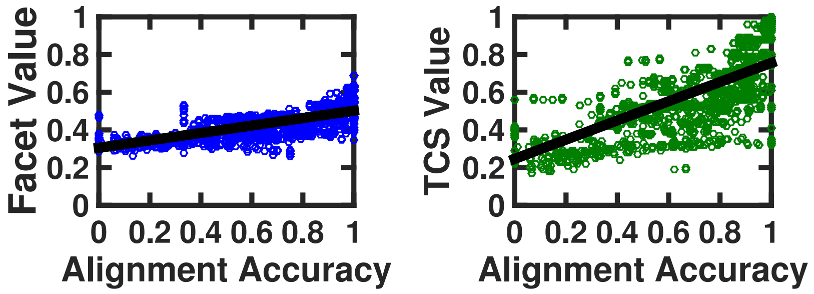

Fig 2. *Relationship of estimators to true accuracy.* Each point in a scatterplot corresponds to an alignment whose true accuracy is on the horizontal axis, and whose value under a given estimator is on the vertical axis. Both scatterplots show the same set of 3,000 alignments (randomly sampled from the more than 209,000 alignments generated by the experiments in Section 3) under the accuracy estimators `Facet` [4] and `TCS` [15].

estimator E. When the feature functions have range $[0, 1]$ and the coefficients form a convex combination, the resulting estimator E will also have range $[0, 1]$. `Facet` uses a collection of five feature functions, many of which make use of predicted secondary structure for the protein sequences [4]. Figure 2 shows the relationship to true accuracy of both the `Facet` and `TCS` [15] estimators.

A parameter advisor uses the estimator to effectively rank alignments, so an estimator just needs to be monotonic in true accuracy. The *difference-fitting* approach learns the coefficients of an estimator that is close to monotonic by fitting the estimator to differences in true accuracy for pairs of training alignments.

Let function $F(A)$ give the *true accuracy* of alignment A, and set $\mathcal{P}$ be a collection of ordered pairs of alignments from training data, where every pair $(A, B) \in \mathcal{P}$ satsifies $F(A) < F(B)$. Difference fitting tries to find an estimator E that increases at least as much as accuracy F on the pairs in $\mathcal{P}$, by minimizing the amount that E falls short. Formally, we find the estimator E^* given by the vector of coefficients $c^* \in \mathcal{R}^d$ that minimizes

$$\sum_{(A,B)\,\in\,\mathcal{P}} w_{AB}\ \max\Big\{\Big(F(B) - F(A)\Big)\ -\ \Big(E(B) - E(A)\Big),\ 0\Big\},$$

where w_{AB} weights the above error for a pair (A, B). Finding the optimal coefficients c^* can be reduced to solving a linear programming problem as follows.

**** EDIT THIS: All pairs from a particular benchmark such that F(A)¿F(B)+0.005 limit of 30,000 per benchmark weight pairs evenly within benchmark, then across benchmark bins.

The linear program has a variable c_i for each estimator coefficient, and an error variable e_{AB} for each pair $(A, B) \in \mathcal{P}$. The constraints are $c_i \geq 0$ and $\sum_i c_i = 1$, which ensure the coefficients form a convex combination, together with $e_{AB} \geq 0$, and

$$e_{AB}\ \geq\ \Big(F(B)\ -\ F(A)\Big)\ -\ \Big(E(B)\ -\ E(A)\Big).$$

(Note that the expressions $E(A)$ and $E(B)$ are linear in the variables c_i, while the quantities $F(A)$ and $F(B)$ are constants.) The linear program then minimizes the objective function $\sum_{(A,B)\in\mathcal{P}} w_{AB}\ e_{AB}$.

For the linear program to be of manageable size for a large number of training alignments, the set $\mathcal{P}$ of pairs must be quite sparse. Kececioglu and DeBlasio [4] describe how to find a good sparse set $\mathcal{P}$ together with a good set of weights w_{AB} by an efficient graph algorithm.

Learning an accuracy estimator with d feature functions using a set $\mathcal{P}$ of p pairs, involves solving the above linear program with $p + d$ variables and $\Theta(p + d)$ inequalities. Evaluating the `Facet` estimator on an alignment with m sequences and n columns, after secondary structure has been predicted for the protein sequences, takes $\Theta(m^2 n)$ time.

2.1.3 Learning an advisor set

DeBlasio and Kececioglu [8, 17] present an efficient approximation algorithm for learning a near-optimal set of parameter choices for an advisor that uses a given estimator. The approximation algorithm follows a greedy strategy, so we call the sets found by the approximation algorithm *greedy sets*, in contrast to *exact sets* that are optimal for the training data, and which can be found by exhaustive search for small instances. These greedy sets tend to generalize better than exact sets, with the remarkable behavior that the greedy sets often outperform exact sets on testing data [8].

The problem of learning an optimal set P of parameter choices for an advisor is formulated as follows. Let U be the *universe* of possible parameter choices that might be included in advisor set P. (Section 2.2 describes how we construct the universe U for aligner advising.) The training data is a collection of *reference alignments* R_i, one for each benchmark B_i, and a collection of *alternate alignments* A_{ij}, where each alignment A_{ij} is computed by running the aligner on the sequences in benchmark B_i using parameter choice $j \in U$. By comparing each alternate alignment to the reference alignment for its benchmark, we can measure the *true accuracy* a_{ij} of each alignment A_{ij}.

For a candidate set $P \subseteq U$ of parameter choices for an advisor that uses estimator E, the set of parameter choices from P that could potentially be *output* by the advisor on benchmark B_i is

$$\mathcal{O}_i(P) \;=\; \operatorname*{argmax}_{j \,\in\, P} \Big\{ E(A_{ij}) \Big\} ,$$

where the argmax gives the set of parameter choices $j \in P$ that are tied for maximizing the estimator E on the benchmark. The advisor could output any parameter choice from $\mathcal{O}_i(P)$, as all of them appear equally good under the estimator, so we assume the advisor selects a choice uniformly at random from this set. Then the *expected accuracy* achieved by the advisor on benchmark B_i using parameter set P is

$$\mathcal{A}_i(P) \;=\; \frac{1}{\big|\mathcal{O}_i(P)\big|} \sum_{j \,\in\, \mathcal{O}_i(P)} a_{ij} ,$$

where again a_{ij} is the true accuracy of alignment A_{ij}.

In learning an advisor set P, we seek a set P that maximizes the advisor's expected accuracy $\mathcal{A}_i(P)$ on the training benchmarks B_i. Formally, we want a set P that maximizes the objective function

$$f(P) \;=\; \sum_i w_i \, \mathcal{A}_i(P) ,$$

where i indexes the benchmarks, and w_i is the weight placed on benchmark B_i. (The benchmark weights correct for sampling bias in the training data, as discussed in Section 3.) In words, we want to find an advisor set $P \subseteq U$ that maximizes the expected accuracy of the parameter choices selected by the advisor, averaged across weighted training benchmarks.

DeBlasio and Kececioglu [8] prove that for a given bound k on the size of advisor set P, finding an optimal set $P \subseteq U$ with $|P| \leq k$ that maximizes objective $f(P)$ is NP-complete.

Greedy sets While this NP-completeness result implies it is unlikely we can efficiently find an *optimal* advisor set, there is a natural greedy algorithm that is guaranteed to efficiently find a *near-optimal* set. For any constant ℓ, the optimal advisor set of cardinality at most ℓ can be found in polynomial time by exhaustive search. The following procedure `Greedy` builds on this idea to find a near-optimal advisor set for cardinality bound k, by starting with an optimal set of size at most ℓ, where $\ell \leq k$, and greedily augmenting it.

```
procedure Greedy($k, \ell$) begin
    Find an optimal subset $P \subseteq U$ of size $|P| \leq \ell$
        that maximizes $f(P)$.
    $\widetilde{P}$ := $P$
    $\widetilde{\ell}$ := $|P|$
    for cardinalities $\widetilde{\ell}+1, \ldots, k$ do begin
        Find parameter choice $j^* \in U - \widetilde{P}$ that
            maximizes $f(\widetilde{P} \cup \{j^*\})$.
        $\widetilde{P}$ := $\widetilde{P} \cup \{j^*\}$
        if $f(\widetilde{P}) \geq f(P)$ then
            $P$ := $\widetilde{P}$
    end
    output $P$
end
```

DeBlasio and Kececioglu [8] prove that procedure `Greedy` is an (ℓ/k)-approximation algorithm for finding an optimal advisor set, for any constant ℓ with $\ell \leq k$.

Learning a greedy advisor set for cardinality bound k when $\ell = 1$, which is the value we use in practice, for the `Facet` estimator on a universe of u parameter choices and a training set of t benchmarks, takes $\Theta(k^2 ut)$ time.

Oracle sets A useful notion in parameter advising introduced by Wheeler and Kececioglu [1] is the concept of an oracle, which is a perfect advisor that has access to the true accuracy of an alignment. For a given advisor set P, an *oracle* selects parameter choice $\text{argmax}_{p \in P}\{F(A_p)\}$, where again function F gives the true accuracy of an alignment. (Equivalently, an oracle is an advisor that uses the perfect estimator F.) An oracle always picks the parameter choice that yields the highest accuracy alignment.

While an oracle is impossible to construct in practice, it gives a theoretical limit on the accuracy achievable by advising with a given set. Furthermore, if we can find the optimal advisor set for an oracle for a given cardinality bound k, which we call an *oracle set*, then the performance of an oracle on an oracle set gives a theoretical limit on how well advising can perform for a given bound k on the number of parameter choices.

Kececioglu and DeBlasio [4] show that finding an oracle set is NP-complete, and give the following integer linear program for finding an optimal oracle set. In our prior notation, an optimal oracle set for cardinality bound k is a set $P \subseteq U$ with $|P| \leq k$ that maximizes $\sum_i w_i \max_{j \in P} a_{ij}$. To formulate this as an integer linear programming problem, let y_j for all $j \in U$, and x_{ij} for all benchmarks B_i and all $j \in U$, be integer variables that either have the value 0 or 1, where $y_j = 1$ iff parameter choice j is selected for oracle set P, and $x_{ij} = 1$ iff choice j is used by the oracle on benchmark B_i. The constraints are $0 \leq x_{ij} \leq y_j \leq 1$, $\sum_j y_j \leq k$, and for each benchmark B_i the constraint $\sum_j x_{ij} = 1$. The objective function is to maximize $\sum_i w_i \sum_j x_{ij}\, a_{ij}$. An optimal solution to this integer linear program gives an optimal oracle set $P^* = \{j \in U : y_j = 1\}$.

Learning an optimal oracle set of cardinality k, for a universe of u parameter choices and a training set of t benchmarks, involves solving the above integer linear program with $\Theta(ut)$ variables and $\Theta(ut)$ inequalities. Kececioglu and DeBlasio [4] show that using the `CPLEX` integer linear programming solver, this formulation permits finding optimal oracle sets in practice even for cardinalities up to $k = 25$.

2.2 Constructing the universe for aligner advising

We extend *parameter advising* with a single aligner to *aligner advising* with a collection of aligners, by having the choices in the advisor set now specify both a particular aligner and a parameter setting for that aligner. To specify the *universe* that such an advisor set is drawn from during learning, we must determine what aligners to consider, and what parameter settings to consider for those aligners.

2.2.1 Determining the universe of aligners

For *default aligner advising*, where the advisor set consists of distinct aligners, each using their default parameter setting, we learned advisor sets over a universe containing as many of the commonly-used aligners from the literature as possible. Specifically, the universe for default advising consisted of the following 17 aligners: `Clustal` [18], `Clustal2` [19], `Clustal Omega` [20], `DIALIGN` [21], `FSA` [22], `Kalign` [23], `MAFFT` [13], `MUMMALS` [3], `MUSCLE` [24], `MSAProbs` [25], `Opal` [1], `POA` [26], `PRANK` [27], `Probalign` [28], `ProbCons` [29], `SATé` [30], and `T-Coffee` [31].

2.2.2 Determining the universe of parameter settings

For *general aligner advising*, we created an parameter universe for each of the aligners we used for default aligner advising, the full list of parameters used is found in Table 1. For each aligner, we enumerated parameter settings by forming a cross product of values for each of its tunable parameters. We determined the values for each tunable parameter by one of two ways. For aligners with web-server versions (namely `Clustal Omega` and `ProbCons`), we used all values recommended for each parameter. For all other aligners, we chose either one or two values above and below the default value for each parameter, to attain a cross product with less than 200 parameter settings. If a range was specified for a numeric parameter, values were chosen to cover this range as evenly as possible. For non-numeric parameters, we used all available options.

Table 1. *Universe of Parameter Settings for General Aligner Advising*

Aligner	Parameter settings	Tunable parameters	Version	Parameter name	Default value, v	Alternate values
ClustalW2 [19]	162	5	2.1	Substitution matrix	GONNET	PAM, BLSM
				Gap open penalty	10	5, 20
				Gap extension penalty	0.2	0.1,0,4
				Gap distance	4	2,8
				End gaps	Off	On
Clustal Omega [20]	120[a]	5	1.2.0	Number of guide tree iterations	0	1, 3, 5
				Number of HMM iterations	0	1, 3, 5
				Number of combined iterations	0	1, 3, 5
				Distance matrix calculations, initial	mBed	Full alignments
				Distance matrix calculations, iterations	mBed	Full alignments
Disalign-T [32]	162	5	0.2.2	Substitution matrix	BLSM62	BLSM75, BLSM90
				Overlap	0	1
				Global Minimum	40	20,60
				Threshold	4	2,6
				Even threshold	4	2,6
Disalign-TX [21]	162	5	1.0.2	Substitution matrix	BLSM62	BLSM75, BLSM90
				Probability Distribution	BLOSUM.diag_prob_t10	BLOSUM75.diag_prob_t2
				Sensitivity	0	1,2
				Threshold	4	2,6
				Even threshold	4	2,6
FSA [22]	200	5	1.15.3	Estimate indel probabilities	On	Off
				Estimate emission probabilities	On	Off
				Regularize learned emission and gap probabilities	On	Off
				Maximum number of iterations of EM	3	1,5,10, 20
				Minimum fractional increase per iteration	0.1	0.01,0.05,0.15,0.2
Kalign [23]	162	4	2.04	Gap open penalty	55	40, 70
				Gap extension penalty	8.5	7, 10
				Terminal gap penalty	4.25	3.5, 5
				Bonus	No	Yes
MAFFT [13]	175	3	6.923b	Substitution matrix	BLSM62	BLSM30, BLSM45, BLSM80, VTML120, VTML200, VTML350
				Gap open penalty	1.53	$\frac{1}{4}v,\ \frac{1}{2}v,\ \frac{3}{2}v,\ 2v$
				Gap extension penalty	0.123	$\frac{1}{2}v,\ 2v,\ 4v$
MSAProbs [25]	175	3	0.9.7	Passes of consistency transformation	2	0,1,3,4,5
				Passes of iterative-refinement	10	0,2,5,20,50,75,100
MUSCLE [24]	160	3	3.8.31	Profile score	Log-expectation: VTML240	Sum-of-pairs: PAM200, VTML240
				Objective function[b]	spm	dp, ps, sp, spf, xp
				Gap open penalty, profile dependent	$\gamma = v$[c]	$\frac{1}{2}v,\ \frac{3}{4}v,\ \frac{5}{4}v,\ \frac{3}{2}v$
				Gap extension penalty	γ	$\frac{1}{2}\gamma$
MUMMALS [3]	29	3[d]	1.01	Differentiate match states in unaligned regions	Yes	No
				Solvent accessibility categories	1	2, 3
				Secondary structure types	3	1
Opal [1]	162	5	3.0b	Substitution matrix	VTML200[e]	BLSM62[e], VTML40[e]
				Internal gap open penalty	$\gamma = 45$	70, 95
				Terminal gap open penalty	0.4γ	$0.05\gamma,\ 0.75\gamma$
				Internal gap extension penalty	$\lambda = 42$	40, 45
				Terminal gap extension penalty	$\lambda - 3$	λ
POA [26]	144	2	1.0.0	Substitution matrix	BLSM80	BLSM62, VTML120, VTML200
				Gap penalty 1	12	3, 24
				Gap penalty 2	6	1, 3
				Progressive alignment	Yes	No
				Global alignment	Yes	No
PRANK [27]	165	3	.140603	Gap rate	0.005	$\frac{1}{5}v,\ \frac{1}{2}v,\ \frac{3}{2}v,\ 2v$
				Gap extension	0.5	$\frac{1}{5}v,\ \frac{1}{2}v,\ \frac{3}{2}v,\ 2v$
				Terminal gaps	Alternate scoring	Normal scoring
				Force insertions to be always skipped	Yes	No
				Iterations	5	1
ProbCons [29]	168[f]	3	1.4	Consistency repetitions	2	0, 1, 3, 4, 5
				Iterative refinement repetitions	100	0, 500, 1000
				Pre-training repetitions	0	1, 2, 3, 4, 5, 20
Probalign [28]	124	3	1.12	Thermodynamic temperature	5	1, 3, 5, 10
				Gap open	22	1, 11, 33, 55
				Gap extension	1	0.25, 0.5, 1.5, 3
T-Coffee [31]	180	3	10.00.r1613	Substitution matrix	BLSM62	BLSM40, BLSM80, PAM120, PAM250, PAM350
				Gap open	0	-50, -500, -1000, -5000
				Gap extension	0	-1, -3, -5, -7, -10
Total	2338					

[a]Parameter settings retrieved from the `Clustal Omega` web-server at EBI (`www.ebi.ac.uk/Tools/msa/clustalo`).

[b]`sp`: sum-of-pairs score; `spf`: dimer approximation of sum-of-pairs score; `spm`: input dependent (`sp` if input is less than 100 sequences, `spf` otherwise); `dp`: dynamic programming score; `ps`: average profile sequence score; `xp`: cross profile score.

[c]Default values for the gap open penalty are -2.9 when the log-expectation profile is chosen, -1439 for sum-of-pairs using `PAM200`, and -300 for sum-of-pairs using `VTML240`. Alternate values are multiples of this default value.

[d]`MUMMALS` is distributed with 29 precomputed hidden Markov models, each of which is associated with a setting of three tunable parameters.

[e]The substitution matrices used by `Opal` are shifted, scaled, and rounded to integer values in the range $[0, 100]$.

[f]Parameter settings retrieved from the `ProbCons` web-server at Stanford (`probcons.stanford.edu`).

3 Results and Discussion

We evaluate the performance of advising through experiments on a collection of protein multiple sequence alignment benchmarks. A full description of the benchmark collection is given in [4], and is briefly summarized below. The experiments compare the accuracy of parameter and aligner advising to the accuracy of individual aligners using their default parameter settings.

The benchmark suites used in our experiments consist of reference alignments that are largely induced by performing structural alignment of the known three-dimensional structures of the proteins. Specifically, we use the `BENCH` suite of Edgar [33], supplemented by a selection of benchmarks from the `PALI` suite [34]. The entire benchmark collection consists of 861 reference alignments.

As is common in benchmark suites, easy-to-align benchmarks are highly over-represented in this collection, compared to hard-to-align benchmarks. To correct for this bias when evaluating average advising accuracy, we binned the 861 benchmarks in our collection by *difficulty*, where the difficulty of a benchmark is its average accuracy under three commonly-used aligners, namely `Clustal Omega`, `MAFFT`, and `ProbCons`, using their default parameter settings. We then divided the full range $[0, 1]$ of accuracies into 10 bins with difficulties $[(j-1)/10, j/10]$ for $j = 1, \ldots, 10$. The weight w_i of benchmark B_i falling in bin j that we used for training is $w_i = (1/10)(1/n_j)$, where n_j is the number of benchmarks in bin j. These weights w_i are such that each difficulty bin contributes equally to the advising objective function $f(P)$. Note that with this weighting, an aligner that on every benchmark gets an accuracy equal to its difficulty, will achieve an average advising accuracy of roughly 50%.

3.1 Parameter advising

We first examine the results of parameter advising for a single aligner using the `Facet` estimator. We learned the coefficients for `Facet` by difference fitting on computed alignments obtained using the oracle set of cardinality $k = 17$ found for the parameter universe for each aligner. (We trained the estimator on an oracle set of this cardinality to match the size of the universe for default aligner advising.) Given this estimator, we constructed greedy advisor sets for each aligner.

Figure 3 shows the accuracy of parameter advising using greedy advisor sets of cardinality $k \leq 15$, for each of the 10 aligners in Table 1, under 12-fold cross-validation. The plot shows advising accuracy on the testing data, averaged over all benchmarks and folds.

Almost all aligners benefit from parameter advising, though their advising accuracy eventually reaches a plateau. While our prior work [8] showed that parameter advising boosts the accuracy of the `Opal` aligner, Figure 3 shows this result is not aligner dependent.

3.2 Aligner advising

To evaluate aligner advising, we followed a similar approach, constructing an oracle set of cardinality $k = 17$ over the union of the universe for default advising from Section 2.2.1 and the universe for general advising from Section 2.2.2, learning coefficients for `Facet` using difference fitting, and constructing greedy sets using `Facet` for default and general advising.

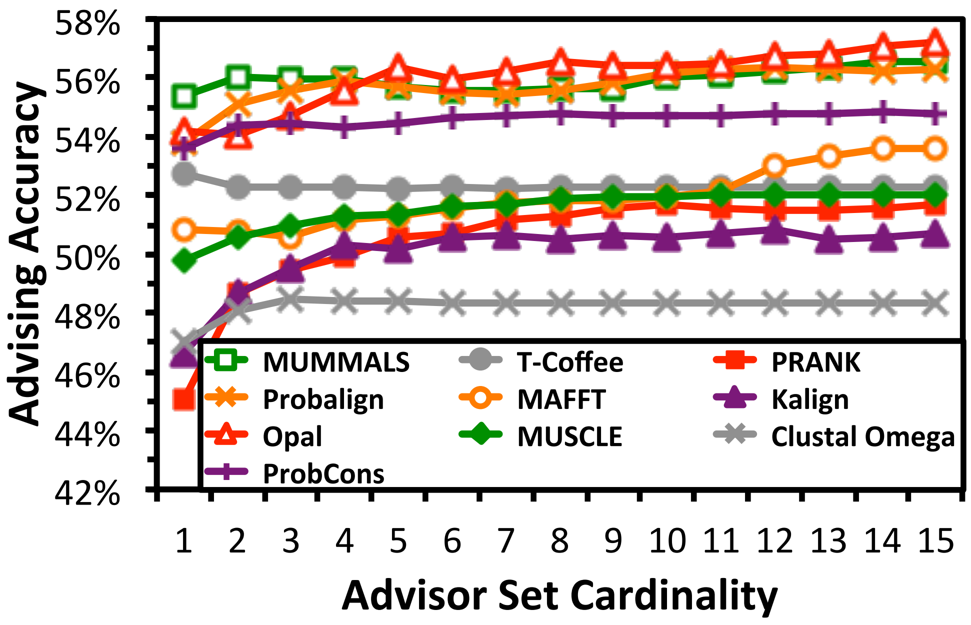

Fig 3. *Accuracy of parameter advising using* `Facet`. The plot shows advising accuracy for each aligner from Table 1, using parameter advising on greedy sets with the `Facet` estimator learned by difference fitting. The horizontal axis is the cardinality of the advisor set, and the vertical axis is the advising accuracy on testing data averaged over all benchmarks and folds, under 12-fold cross-validation.

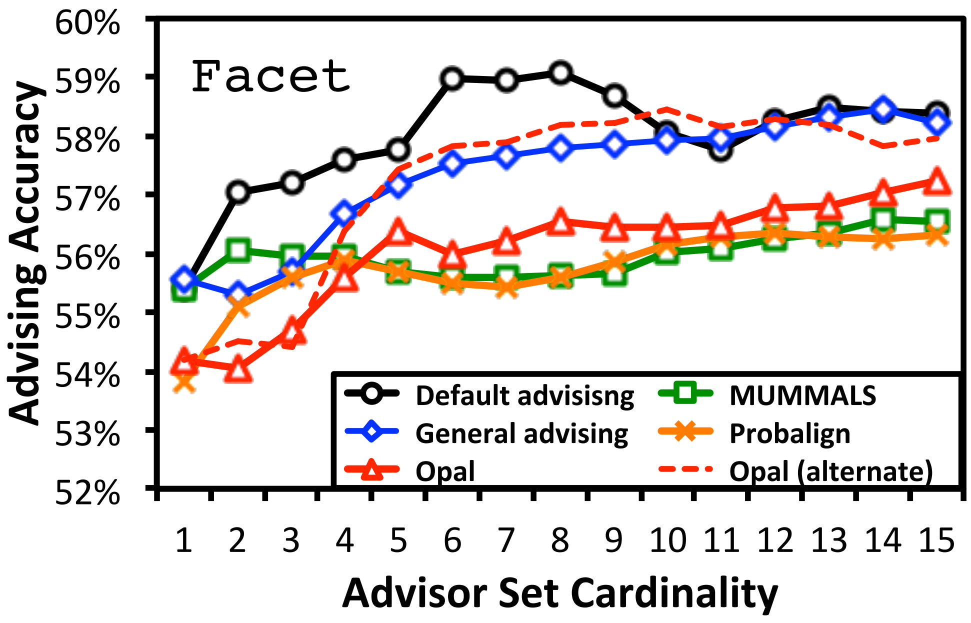

Fig 4. *Aligner advising and parameter advising using* `Facet`. The plot shows default and general aligner advising accuracy, and parameter advising accuracy for `Opal`, `MUMMALS`, and `Probalign`, using the `Facet` estimator. The horizontal axis is the cardinality of the advisor set, and the vertical axis is advising accuracy on testing data averaged over all benchmarks and folds under 12-fold cross-validation.

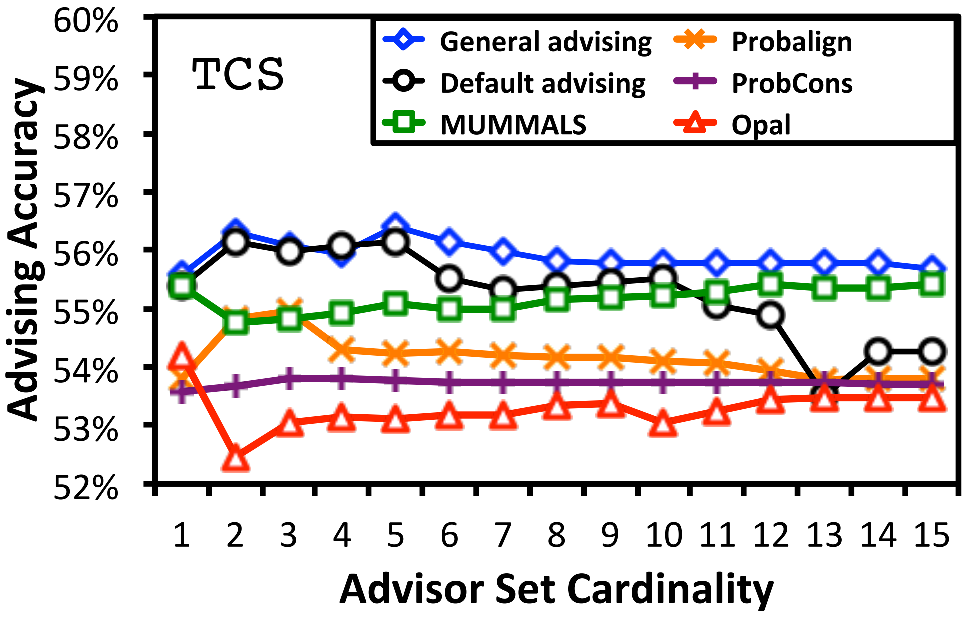

Fig 5. *Aligner advising and parameter advising using* `TCS`. The plot shows default and general aligner advising accuracy, and parameter advising accuracy for `Opal`, `MUMMALS`, `Probalign`, and `ProbCons`, using the `TCS` estimator. The horizontal axis is the cardinality of the advisor set, and the vertical axis is advising accuracy on testing data averaged over all benchmarks and folds under 12-fold cross-validation.

Figure 4 shows the accuracy of default and general advising using greedy sets of cardinality $k \leq 15$, along with the three best parameter advising curves from Figure 3, for `Opal`, `Probalign`, and `MUMMALS`. The plot shows advising accuracy on testing data, averaged over benchmarks and folds.

The dashed red curve in Figure 4 also shows the accuracy of `Opal` for parameter advising with greedy sets computed over an alternate universe of much more fine-grained parameter choices. To construct this alternate universe, we first started with a set of over 16,000 parameter settings, and for each training fold chose the 15 parameters with highest average accuracy on each difficulty bin. Unioning these choices across the bins, and removing duplicates, gave a universe of 147 to 150 settings for each fold. Note that the dashed curve for parameter advising with `Opal`, using greedy sets from these finer universes for each fold, essentially matches the accuracy of general advising at cardinality $k \geq 4$.

3.2.1 Testing the significance of improvement

To test the statistical significance of the improvement in default advising accuracy over using a single default aligner, we used the one-tailed Wilcoxon sign test [35]. Performing this test in each difficulty bin, we found a significant improvement in accuracy ($p < 0.05$) on benchmarks with difficulty $(0.3, 0.4]$ at all cardinalities $2 \leq k \leq 15$, and on benchmarks with difficulty at most 0.4 at cardinality $6 \leq k \leq 9$.

We also tested the significance of the improvement of default advising over the best parameter advisor at each cardinality k (namely `MUMMALS` for $k \leq 4$ and `Opal` for $k \geq 5$), and found that at cardinality $k \geq 5$ there is again significant improvement ($p < 0.05$) on benchmarks with difficulty $(0.3, 0.4]$.

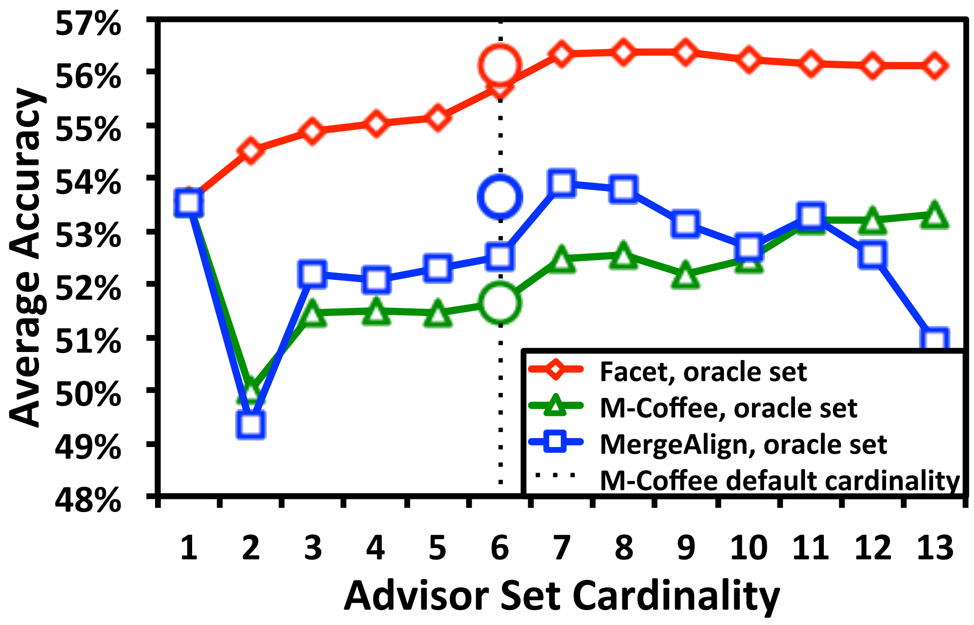

Fig 6. *Accuracy of aligner advising compared to* `M-Coffee`. The plot shows average accuracy for aligner advising using `Facet`, and meta-alignment using `M-Coffee`, on oracle sets of aligners. Performance on the default `M-Coffee` set of six aligners is indicated by large circles on the dotted vertical line. The horizontal axis is cardinality of the oracle sets, and the vertical axis is average accuracy on testing data over all benchmarks and folds under 12-fold cross-validation.

3.2.2 Advising with an alternate estimator

We also evaluated in the same way parameter advising and aligner advising on greedy sets using the `TCS` estimator [15] (the best other estimator for advising from the literature). Figure 5 shows results using `TCS` for parameter advising (on the four most accurate aligners), and for general and default aligner advising. Note that while `TCS` is sometimes able to increase accuracy above using a single default parameter, this increase is smaller than for `Facet`; moreover, `TCS` often has a decreasing trend in accuracy for increasing cardinality.

3.3 Comparing ensemble alignment to meta-alignment

Another approach to combining aligners is the so-called *meta-alignment* approach of `M-Coffee` [9] (described in Section 1). `M-Coffee` computes a multiple alignment using position-dependent substitution scores obtained from alternate alignments generated by a collection of aligners. By default, `M-Coffee` uses the following six aligners: `Clustal2`, `T-Coffee`, `POA`, `MUSCLE`, `MAFFT`, `Dialign-T` [32], `PCMA` [36], and `ProbCons`. The tool also allows use of `Clustal`, `Clustal Omega`, `Kalign`, `AMAP` [37], and `Dialign-TX`. Figure 6 shows the average accuracy of both `M-Coffee` and our ensemble approach with `Facet`, using the default aligner set of `M-Coffee` (the dotted vertical line with large circles), as well as oracle sets constructed over this `M-Coffee` universe of 13 aligners.

3.4 Advising accuracy within difficulty bins

Figure 7 shows advising accuracy within difficulty bins for default aligner advising compared to using the default parameter settings for the three aligners with highest

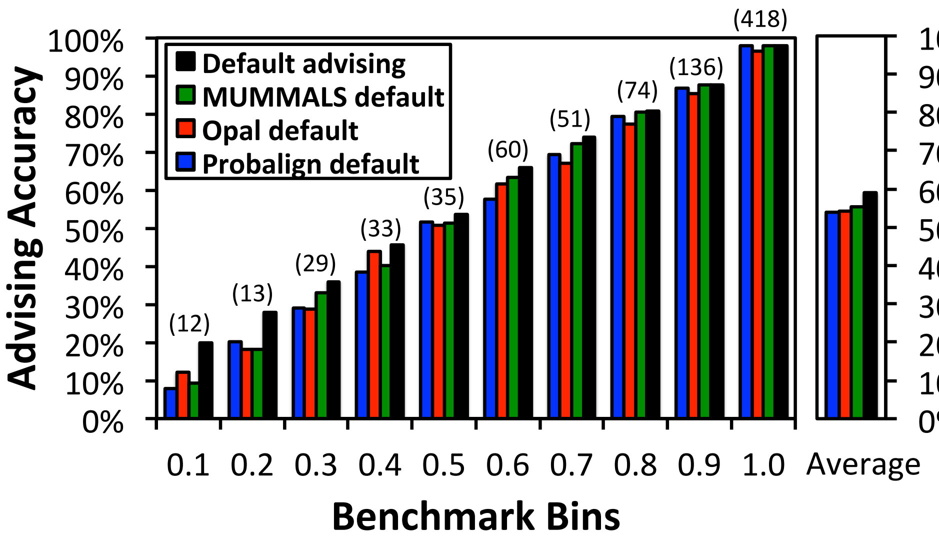

Fig 7. *Accuracy of default aligner advising, and aligners with their default settings, within difficulty bins.* In the bar chart on the left, the horizontal axis shows all ten benchmark bins, and the vertical bars show accuracy averaged over just the benchmarks in each bin. The accuracy of default advising using the `Facet` estimator is shown for the greedy sets of cardinality $k = 5$, along with the accuracy of the default settings for `Probalign`, `Opal`, and `MUMMALS`. The bar chart on the right shows accuracy uniformly averaged over the bins. In parentheses above the bars are the number of benchmarks in each bin.

average accuracy, namely `MUMMALS`, `Opal`, and `Probalign`. The figure displays the default advising result from Section 3.2 at cardinality $k = 5$. The bars in the chart show average accuracy over the benchmarks in each difficulty bin, as well as the average accuracy across all bins. (The number of benchmarks in each bin is in parentheses above the bars.) Note that aligner advising gives the greatest boost for the hardest-to-align benchmarks: for the bottom two bins, advising yields an 8% increase in accuracy over the best aligner using its default parameter setting.

3.5 Generalization of aligner advising

The results thus far have shown advising accuracy averaged over the testing data associated with each fold. We now compare the training and testing advising accuracy to assess how our method might generalize to data not in our benchmark set.

Figure 8 shows the average accuracy of default and general aligner advising on both training and testing data. Note that the drop between training and testing accuracy is much larger for general advising than for default advising, resulting in general advising performing worse than default advising though its training accuracy is much higher. This indicates that general advising is strongly overfitting to the training data, but could potentially achieve much higher testing accuracy. Additionally, there is a drop in training accuracy for default advising with increasing cardinality, though after its peak an advisor using greedy sets should remain flat in training accuracy as cardinality increases when using a strong estimator.

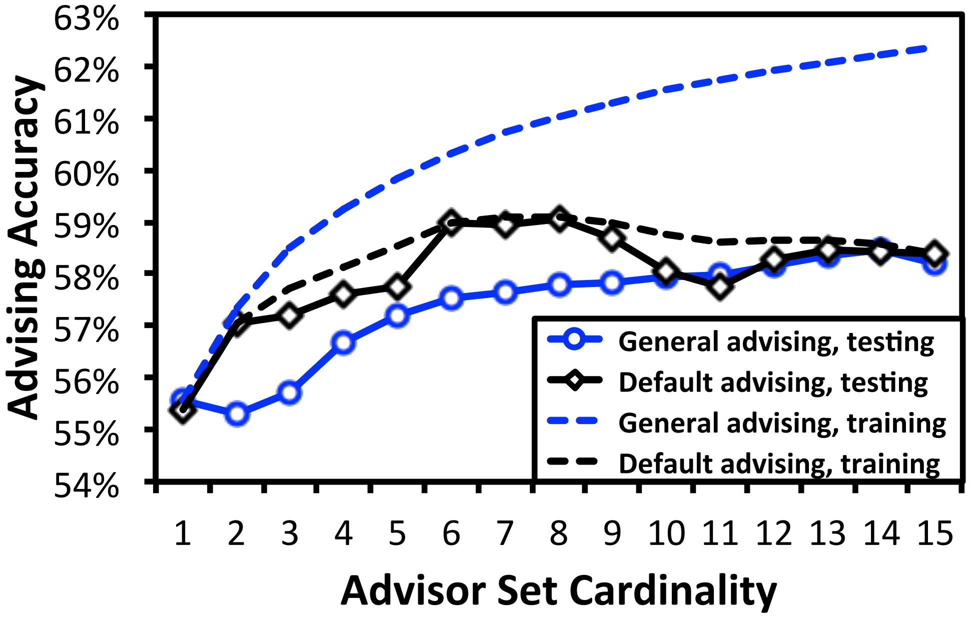

Fig 8. *General and default aligner advising on training and testing data.* The plot shows general and default aligner advising accuracy using `Facet`. Accuracy on the training data is shown with dashed lines, and on the testing data with solid lines. The horizontal axis is cardinality of the advisor set, and the vertical axis is advising accuracy averaged over all benchmarks and folds under 12-fold cross-validation.

3.6 Theoretical limit on advising accuracy

An *oracle* is an advisor that uses a perfect estimator, always choosing the alignment from a set that has highest true accuracy. To examine the theoretical limit on how well aligner advising can perform, we compare the accuracy of aligner advising using `Facet` with the performance of an oracle. Figure 9 shows the accuracy of both default and general aligner advising using greedy sets, as well as the performance of an oracle using oracle sets computed on the default and general advising universes. (Recall an *oracle set* is an optimal advisor set for an oracle.) The plot shows advising accuracy on testing data, averaged over all benchmarks and folds. The large gap in performance between the oracle and an advisor using `Facet` shows the increase in accuracy that could potentially be achieved by developing an improved estimator.

3.7 Composition of advisor sets

Table 2 lists the greedy advisor sets for both default and general advising for all cardinalities $k \leq 10$. A consequence of the construction of greedy advisor sets is that the greedy set of cardinality k consists of the entries in a column in the first k rows of the table. The table shows these sets for just one fold from the 12-fold cross-validation. For general advising sets, an entry specifies the aligner that is used, and for aligners from the general advising universe, a tuple of parameter values in the order listed in Table 1. The two exceptions are `MUMMALS`, whose 6-tuple comes from its predefined settings file, and whose last three elements correspond to the three parameters listed in Table 1; and `MSAProbs`, whose empty tuple stands for its default setting. It is interesting that other than `MSAProbs`, the general advising set does not contain any aligner's default parameter settings, though its values are close to the default setting.

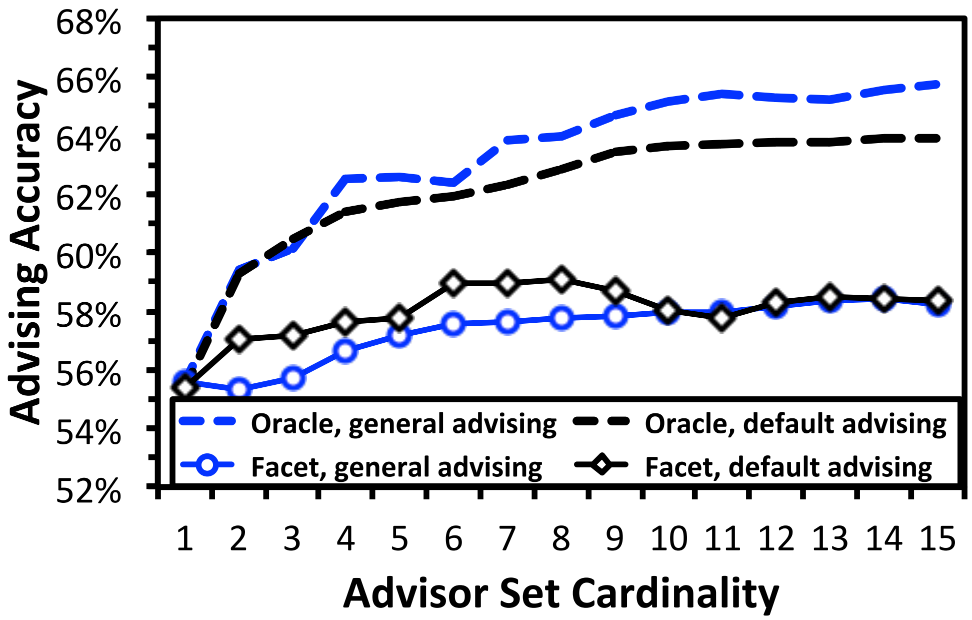

Fig 9. *Accuracy of aligner advising using a perfect estimator.* The plot shows advising accuracy for default and general aligner advising, both on oracle sets for a perfect estimator, and on greedy sets for the `Facet` estimator. The horizontal axis is the cardinality of the advisor set, and the vertical axis is advising accuracy on testing data averaged over all benchmarks and folds under 12-fold cross-validation.

Table 2. *Greedy Default and General Advising Sets*

	Default advising	General advising
1	`MUMMALS`	`MUMMALS` (0.2, 0.4, 0.6, 1, 2, 3)
2	`Opal`	`Opal` (VTML200, 45, 2, 45, 45)
3	`Probalign`	`Opal` (BLSM62, 70, 3, 45, 42)
4	`Kalign`	`MUMMALS` (0.15, 0.2, 0.6, 1, 1, 3)
5	`MUSCLE`	`Opal` (BLSM62, 45, 33, 42, 42)
6	`T-Coffee`	`MSAProbs` ()
7	`PRANK`	`Kalign` (55, 8.5, 4.25, 0)
8	`Clustal Omega`	`MAFFT` (VTML200, 0.7515, 0.492)
9	`DIALIGN`	`Opal` (BLSM62, 95, 4, 45, 42)
10	`ProbCons`	`Opal` (BLSM62, 45, 2, 45, 42)

3.8 Running time for advising

We compared the time to evaluate the `Facet` estimator on an alignment to the time needed to compute that alignment by the three aligners used for determining alignment difficulty: `Clustal Omega`, `MAFFT`, and `ProbCons`. To compute the average running time for these aligners on a benchmark, we measured the total time for each of these aligners to align all 861 benchmarks on a desktop computer with a 2.4 GHz Intel i7 8-core processor and 8 Gb of RAM. The average running time for `Clustal Omega`, `MAFFT`, and `ProbCons` was less than 1 second per benchmark, as was the average running time for `Facet`. As stated in Section 2.1.2, the time complexity for `Facet` is linear in the number of columns in an alignment, and should take relatively less time than computing an alignment for benchmarks with long sequences; the standard benchmark suites tend to include short sequences, however, which are fast to align. This time to evaluate `Facet` does not include the time to predict protein secondary structure, which is done once for the sequences in a benchmark, and was performed using `PSIPRED` [38] version `3.2` with its standard settings. Secondary structure prediction with a tool like `PSIPRED` has a considerably longer running time than alignment, due to an internal `PSI-BLAST` search during prediction; on average, `PSIPRED` took just under 6 minutes per benchmark to predict secondary structure.

3.9 Further research

An important question left to explore is how to learn advisor sets that have improved *generalization*. While greedy advisor sets for general aligner advising achieve very high accuracy on training data, this does not translate to similar accuracy on testing data due to overfitting. Standard techniques from machine learning for improving generalization like regularization do not apply here, as the number of parameters for each aligner and the number of choices in the advisor set are both fixed. Applying this advising framework to *DNA* and *RNA* sequence alignment also seems fruitful.

4 Acknowledgements

We thank the reviewers for their helpful comments. This work was supported by NSF Grant IIS-1217886 to J.K., and a PhD fellowship to D.D. from the University of Arizona IGERT in Comparative Genomics through US National Science Foundation Grant DGE-0654435. DD was also partially supported at Carnegie Mellon University by NSF Grant CCF-1256087, NSF Grant CCF-131999, NIH Grant R01HG007104, and Gordon and Betty Moore Foundation Grant GBMF4554 all to Carl Kingsford.

References

1. Wheeler TJ, Kececioglu JD. Multiple alignment by aligning alignments. Bioinformatics. 2007;23(13):i559–68.

2. Van Walle I, Lasters I, Wyns L. `SABmark`: a benchmark for sequence alignment that covers the entire known fold space. Bioinformatics. 2005;21(7):1267–1268.

3. Pei J, Grishin NV. `MUMMALS`: multiple sequence alignment improved by using hidden Markov models with local structural information. Nucleic Acids Research. 2006;34(16):4364–4374.

4. Kececioglu JD, DeBlasio DF. Accuracy Estimation and Parameter Advising for Protein Multiple Sequence Alignment. Journal of Computational Biology. 2013;20(4):259–279.

5. DeBlasio DF, Kececioglu JD. `Facet`: software for accuracy estimation of protein multiple sequence alignments; 2014. `facet.cs.arizona.edu`.

6. Wheeler TJ, Kececioglu JD. `Opal`: software for sum-of-pairs multiple sequence alignment; 2012. `opal.cs.arizona.edu`.

7. Zhihua Z. Ensemble Methods: Foundations and Algorithms. Chapman and Hall; 2012.

8. DeBlasio DF, Kececioglu JD. Learning Parameter-Advising Sets for Multiple Sequence Alignment. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2015;.

9. Wallace IM, O'Sullivan O, Higgins DG, Notredame C. `M-Coffee`: combining multiple sequence alignment methods with `T-Coffee`. Nucleic Acids Research. 2006;34(6):1692–1699.

10. Collingridge PW, Kelly S. `MergeAlign`: improving multiple sequence alignment performance by dynamic reconstruction of consensus multiple sequence alignments. BMC Bioinformatics. 2012;13(1):117. doi:10.1186/1471-2105-13-117.

11. Muller J, Creevey CJ, Thompson JD, Arendt D, Bork P. `AQUA`: automated quality improvement for multiple sequence alignments. Bioinformatics. 2010;26(2):263–265.

12. Edgar RC. `MUSCLE`: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics. 2004;5(1):113.

13. Katoh K, Kuma Ki, Toh H, Miyata T. `MAFFT` version 5: improvement in accuracy of multiple sequence alignment. Nucleic Acids Research. 2005;33(2):511–518.

14. Thompson JD, Plewniak F, Ripp R, Thierry JC, Poch O. Towards a reliable objective function for multiple sequence alignments1. Journal of Molecular Biology. 2001;314(4):937–951.

15. Chang JM, Tommaso PD, Notredame C. `TCS`: A new multiple sequence alignment reliability measure to estimate alignment accuracy and improve phylogenetic tree reconstruction. Molecular Biology and Evolution. 2014;.

16. DeBlasio DF, Wheeler TJ, Kececioglu JD. Estimating the accuracy of multiple alignments and its use in parameter advising. Proceedings of the 16th Conference on Research in Computational Molecular Biology *(RECOMB)*. 2012; p. 45–59.

17. DeBlasio DF, Kececioglu JD. Learning parameter sets for alignment advising. Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics *(BCB)*. 2014; p. 230–239.

18. Thompson JD, Higgins DG, Gibson TJ. `ClustalW`: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Research. 1994;22(22):4673–4680.

19. Larkin MA, et al. `ClustalW` and `ClustalX` version 2.0. Bioinformatics. 2007;23(21):2947–2948.

20. Sievers F, et al. Fast, scalable generation of high-quality protein multiple sequence alignments using `Clustal Omega`. Molecular Systems Biology. 2011;7(1):539–539.

21. Subramanian AR, Kaufmann M, Morgenstern B. DIALIGN-TX: greedy and progressive approaches for segment-based multiple sequence alignment. Algorithms for Mol Biology. 2008;3(1):6.

22. Bradley RK, Roberts A, Smoot M, Juvekar S, Do J, Dewey C, et al. Fast Statistical Alignment. PLoS Computational Biology. 2009;5(5):e1000392.

23. Lassmann T, Sonnhammer E. Kalign: an accurate and fast multiple sequence alignment algorithm. BMC Bioinformatics. 2005;6(1):298.

24. Edgar RC. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Research. 2004;32(5):1792–1797.

25. Liu Y, Schmidt B, Maskell DL. MSAProbs: multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities. Bioinformatics. 2010;26(16):1958–1964. doi:10.1093/bioinformatics/btq338.

26. Lee C, Grasso C, Sharlow MF. Multiple sequence alignment using partial order graphs. Bioinformatics. 2002;18(3):452–464.

27. Loytynoja A, Goldman N. An algorithm for progressive multiple alignment of sequences with insertions. Proceedings of the National Academy of Sciences. 2005;102(30):10557–10562.

28. Roshan U, Livesay DR. Probalign: multiple sequence alignment using partition function posterior probabilities. Bioinformatics. 2006;22(22):2715–2721.

29. Do CB, Mahabhashyam MSP, Brudno M, Batzoglou S. ProbCons: probabilistic consistency-based multiple sequence alignment. Genome Research. 2005;15(2):330–340.

30. Liu K, Warnow TJ, Holder MT, Nelesen SM, Yu J, Stamatakis AP, et al. SATé-II: Very Fast and Accurate Simultaneous Estimation of Multiple Sequence Alignments and Phylogenetic Trees. Systematic Biology. 2011;61(1):90–106.

31. Notredame C, Higgins DG, Heringa J. T-Coffee: A novel method for fast and accurate multiple sequence alignment. Journal of Molecular Biology. 2000;302(1):205–217.

32. Subramanian AR, Weyer-Menkhoff J, Kaufmann M, Morgenstern B. DIALIGN-T: An improved algorithm for segment-based multiple sequence alignment. BMC Bioinformatics. 2005;6(1).

33. Edgar RC. BENCH; 2009. drive5.com/bench.

34. Balaji S, Sujatha S, Kumar SSC, Srinivasan N. PALI: a database of Phylogeny and ALIgnment of homologous protein structures. Nucleic Acids Research. 2001;29(1):61.

35. Wilcoxon F. Individual Comparisons by Ranking Methods. Biometrics Bulletin. 1945;1(6):pp. 80–83.

36. Pei J, Sadreyev R, Grishin NV. PCMA: fast and accurate multiple sequence alignment based on profile consistency. Bioinformatics. 2003;19(3):427–428. doi:10.1093/bioinformatics/btg008.

37. S Schwartz A, Pachter L. Multiple alignment by sequence annealing. Bioinformatics. 2007;23(2):e24–e29. doi:10.1093/bioinformatics/btl311.

38. Jones DT. Protein secondary structure prediction based on position-specific scoring matrices. Journal of Molecular Biology. 1999;292(2):195 – 202. doi:10.1006/jmbi.1999.3091.