Learning Parameter-Advising Sets for Multiple Sequence Alignment

Dan DeBlasio and John Kececioglu

Abstract—While the multiple sequence alignment output by an aligner strongly depends on the parameter values used for the alignment scoring function (such as the choice of gap penalties and substitution scores), most users rely on the single default parameter setting provided by the aligner. A different parameter setting, however, might yield a much higher-quality alignment for the specific set of input sequences. The problem of picking a good choice of parameter values for specific input sequences is called *parameter advising*. A parameter advisor has two ingredients: (i) a *set* of parameter choices to select from, and (ii) an *estimator* that provides an estimate of the accuracy of the alignment computed by the aligner using a parameter choice. The parameter advisor picks the parameter choice from the set whose resulting alignment has highest estimated accuracy.

We consider for the first time the problem of learning the optimal set of parameter choices for a parameter advisor that uses a given accuracy estimator. The optimal set is one that maximizes the expected true accuracy of the resulting parameter advisor, averaged over a collection of training data. While we prove that learning an optimal set for an advisor is NP-complete, we show there is a natural approximation algorithm for this problem, and prove a tight bound on its approximation ratio. Experiments with an implementation of this approximation algorithm on biological benchmarks, using various accuracy estimators from the literature, show it finds sets for advisors that are surprisingly close to optimal. Furthermore, the resulting parameter advisors are significantly more accurate in practice than simply aligning with a single default parameter choice.

Index Terms—Multiple sequence alignment, alignment scoring functions, parameter values, accuracy estimation, parameter advising.

1 Introduction

A key issue in multiple sequence alignment not often addressed is the choice of parameter values to use for the alignment scoring function of an aligner. The standard tools for multiple sequence alignment all use alignment scoring functions that have many parameters that must be set, such as the choice of matrix that scores substitutions in the alignment, and the penalties that are charged for gaps in the alignment formed by runs of insertions or deletions. In the face of the multitude of possible settings for these parameters, most users do not vary parameter values when computing an alignment of their sequences, but simply rely on the default parameter choice supplied by the aligner. The multiple alignment computed by an aligner, however, can change radically as parameter values are varied, and a parameter setting other than the default could yield a much higher-quality alignment of the user's particular sequences.

To give a concrete example, Figure 1 shows a set of benchmark protein sequences aligned by the `Opal` aligner [2], [3] under two parameter settings: the optimal default setting, which is the parameter setting that achieves the highest average true accuracy across a suite of alignment benchmarks, and a second non-default setting. (Here a parameter setting is a five-tuple that specifies the substitution scoring matrix and the values of four gap penalties.) This particular non-default parameter setting happens to come from the optimal set of two parameter choices (as discussed in Section 3), and yields a much more accurate alignment of these sequences.

This begs the question, however, of how can a user in practice recognize which of these two alignments is more accurate? In reality, when aligning sequences, the correct alignment is of course not known, so the *true accuracy* of a computed alignment cannot be measured. In this situation, we rely on an *accuracy estimator* that is positively correlated with true accuracy, and we choose the alignment that has higher estimated accuracy. To provide an illustration, Figure 2 shows the correlation with true accuracy of three accuracy estimators from the literature on the same collection of computed alignments.

In the example of Figure 1, under the `Facet` estimator [4], [5], the alignment of higher *true* accuracy does in fact have higher *estimated* accuracy. So a user armed with `Facet` could pick the better parameter choice to use with `Opal` on these input sequences.

Combining these ideas of a set of candidate parameter choices and an accuracy estimator, in an automated procedure, leads to the notion of a *parameter advisor* that recommends a parameter setting for an aligner to use on the given input sequences. In this paper, we study how to learn the set of parameter choices for a parameter advisor, both *theoretically* from the viewpoint of algorithms and complexity, and *practically* from the standpoint of performance of an implementation on real biological data.

1.1 Related work

The notion of parameter advising was introduced in Wheeler and Kececioglu [2] as an often-overlooked stage in multiple sequence alignment, and was first studied in depth

• *D. DeBlasio and J. Kececioglu are with the Department of Computer Science at the University of Arizona, Tucson, AZ 85721, USA. E-mail: {deblasio,kece}@cs.arizona.edu*

A preliminary conference version of this paper appeared as [1]. Corresponding author: Dan DeBlasio.

```
d1gvoa   203 ... gsvenrarlvlevvdavcnewsad-RIGIRVSPigtfqnvdngpnee--adalyl--- ... 255
d2dora   141 ... ydfeatekllke-----vftfftk-PLGVKLPPyf--------------dlvhfdim ... 178
d1oyb    215 ... gsienrarftlevvdalveaighe-KVGLRLSPygvfnsmsggaetgivaqyayvage ... 272
d1o94a1  193 ... gslenrarfwletlekvkhavgsdcAIATRFGV----------------dtvygpgq ... 234
d1ep3a   147 ... tdpevaaalvka-----ckavskv-PLYVKLSPnvt-------------divpiaka ... 185
```

(a) Higher-accuracy alignment, non-default parameter choice

```
d1gvoa   184 ... yl-lhqflspssnqrtdqyggsvenrarlvlevvdavcnewsad-RIGIRVSPigtfq ... 240
d2dora   159 ... kP-LGVKLPPyf--dlvhfdimaeilnqfpltYVNSV-nsig----nglfidpeaesv ... 209
d1oyb    196 ... yl-lnqfldphsntrtdeyggsienrarftlevvdalveaighe-KVGLRLSPygvfn ... 252
d1o94a1  174 ... yl-plqflnpyynkrtdkyggslenrarfwletlekvkhavgsdcAIATRF---GVdt ... 228
d1ep3a   164 ... kvPLYVKLSPnv-tdivpiakaveaagadGLTMIntl--------mgvrfdlktrqp ... 212
```

(b) Lower-accuracy alignment, default parameter choice

Fig. 1. *Parameter choice affects the accuracy of computed alignments.* (a) Part of an alignment of benchmark `sup_155` from the `SABRE` [6] suite computed by `Opal` [2] using non-default parameter choice $(\texttt{VTML200}, 45, 6, 40, 40)$; this alignment has accuracy value 75.8%, and `Facet` [4] estimator value 0.423. (b) Alignment of the same benchmark by `Opal` using the *optimal* default parameter choice $(\texttt{BLSM62}, 65, 9, 44, 43)$; this alignment has lower accuracy 57.3%, and lower `Facet` value 0.402. 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.

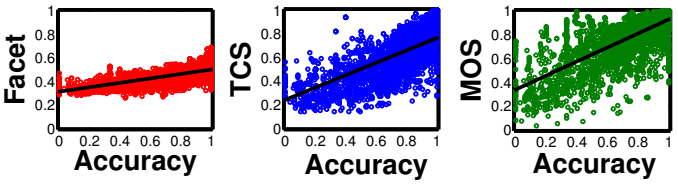

Fig. 2. *Correlation of estimators with true accuracy.* Each point in a scatterplot corresponds to an alignment whose true accuracy is on the horizontal axis, and whose value under a given estimator is on the vertical axis. The scatterplots show the same set of over 4,000 alignments under the accuracy estimators `Facet` [4], `TCS` [11], and `MOS` [12].

in DeBlasio, Wheeler, and Kececioglu [7], both with regard to constructing accuracy estimators, and finding parameter sets for a perfect advisor called an oracle.

Kececioglu and DeBlasio [4] give a broad survey of accuracy estimators from the literature. Briefly, estimators can be categorized as *scoring-function-based* [7], [8], [9], [10], [11], which combine local attributes of an alignment into a score, and *support-based* [12], [13], [14], [15], which assess the quality of an alignment in terms of its support from alternate alignments. Of these estimators, the most accurate for protein alignments are `Facet` (DeBlasio et al. [7]), `TCS` (Chang et al. [11], which supersedes `COFFEE` [8]), `MOS` (Lassmann and Sonnhammer [12]), `PredSP` (Ahola et al. [10]), and `GUIDANCE` (Penn et al. [14]). Kececioglu and DeBlasio [4] compare these estimators, except for `TCS` and `GUIDANCE`, and show that `Facet`, which is a weighted combination of five real-valued feature functions, strongly outperforms these other estimators for the task of parameter advising. Further experiments in this paper show `Facet` outperforms `TCS` and `GUIDANCE` as well.

The emphasis of our prior work [4], [7] is mainly on *accuracy estimation* for parameter advising, resulting in the `Facet` estimator [5]. Our prior work presented a *class of estimators* that are polynomials in alignment feature functions, and gave two techniques for efficiently learning optimal coefficients for these polynomials via linear and quadratic programming. This work introduced new *feature functions* for protein multiple sequence alignments that make use of predicted secondary structure, including a feature called Secondary Structure Blockiness, whose evaluation involves efficiently computing an optimal packing of blocks of common secondary structure. Our prior work also showed that optimal sets of parameter choices for a *perfect advisor* called an oracle (that knows the true accuracy of an alignment) could be found by integer linear programming, which made the computation of optimal oracle sets feasible in practice, even for very large cardinalities.

1.2 Our contributions

In this paper we focus on learning sets of parameter choices for a *realistic advisor*, where these sets are tailored to the actual estimator used by the advisor (as opposed to finding parameter sets for a perfect but unattainable oracle advisor), and we formalize for the first time this *new problem* of learning an optimal parameter set for an imperfect advisor. We prove that while learning such sets is NP-complete, there is an efficient greedy *approximation algorithm* for this learning problem, and we derive a tight bound on its worst-case approximation ratio. Experiments show that the *greedy* parameter sets found by the approximation algorithm for an advisor, that uses `Facet`, `TCS`, `MOS`, `PredSP`, or `GUIDANCE` as its estimator, outperform optimal *oracle* sets at all cardinalities. Furthermore, on the training data, for some estimators these suboptimal greedy sets perform surprisingly close to the optimal *exact* sets found by exhaustive search, and moreover, these greedy sets actually *generalize better* than exact sets. As a consequence, on testing data, for some estimators the greedy sets output by the approximation algorithm can actually give superior performance to exact sets for parameter advising.

1.3 Plan of the paper

In Section 2 we next provide background on estimators and advisors. Section 3 then defines the new problem of learning optimal parameter sets for an advisor. Section 4 presents a greedy approximation algorithm for learning parameter sets, and proves a tight bound on its approximation ratio. Section 5 proves that learning optimal parameter sets is NP-complete. Section 6 presents results from experiments with our learning algorithms on real biological benchmarks. Finally Section 7 gives conclusions and provides directions for further research.

2 Estimators and Advisors

We first briefly review the concepts of accuracy estimators and parameter advisors.

2.1 Accuracy estimation

In the approach of our prior work [4], [7] on estimating the unknown accuracy of an alignment, we assume 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 estimators* E that we consider are linear combinations of these functions, $E(A) = \sum_{1 \le i \le d} c_i \, g_i(A)$, where the coefficents $c_1, \ldots, c_d$ specify the estimator E. The *true accuracy* of an alignment A is usually measured as a real value in the range $[0, 1]$, such as the so-called SPS-score [16], which is

the fraction of substitutions in the core columns of the ground-truth alignment that are recovered by computed alignment A. We assume the feature functions have the range $[0, 1]$; when the coefficients form a convex combination (namely $c_i \geq 0$ and $\sum_i c_i = 1$), the resulting estimator E will then also have the range $[0, 1]$. Our prior work showed that this class of linear estimators $\sum_i c_i\, g_i(A)$ is as general as polynomial estimators: any estimator that is a higher-degree polynomial in the $g_i(A)$ can always be reduced to a linear estimator by appropriately defining new feature functions that are products of the original feature functions.

Given the feature functions g_i, the coefficients of an estimator E can be learned by fitting to true accuracy values on alignment benchmarks for which the "correct" alignment, also called a *reference alignment*, is known. Our prior work [4], [7] presented two techniques for fitting an estimator, called difference fitting and value fitting, and reduced these techniques to linear and quadratic programming.

2.2 Parameter advising

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 p, and then selects the parameter choice p^* that has highest *estimated* accuracy: $p^* = \operatorname{argmax}_{p \in P} E(A_p)$. Since such an 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.

Our prior work [4], [7] presented a technique for finding a small set P that maximizes the true accuracy of a perfect advisor called an oracle. An *oracle* has access to the true accuracy of computed alignments (while an advisor does not, and must rely on an accuracy estimator); accordingly, an oracle always selects the parameter choice from P that has highest *true* accuracy. For a given cardinality k, an *oracle set* is a set S of k parameter choices that maximizes the true accuracy of the alignments output by an oracle using S. Our prior work showed that oracle sets could be found by integer linear programming, and that optimal oracle sets could be computed in practice up to very large cardinalities (for example, even for $k = 25$).

In contrast to finding oracle sets, here we consider how to learn the optimal set P of a given cardinality that maximizes the true accuracy of an imperfect advisor that uses a given *estimator*.

3 Learning Optimal Advisor Sets

We now define the computational problem of learning an optimal set of parameter choices for an advisor using a given accuracy estimator. Throughout we assume the features used by the advisor's estimator are specified and fixed.

From a machine learning perspective, our problem formulation seeks an advisor with optimal accuracy on a collection of training data. The underlying training data is

- a suite of *benchmarks*, where each benchmark B_i in the suite consists of a set of sequences to align, together with a *reference alignment* R_i for these sequences that represents their "correct" alignment, and
- a collection of *alternate alignments* of these benchmarks, where each alternate alignment A_{ij} results from aligning the sequences in benchmark i using a parameter choice j that is drawn from a given universe U of parameter choices.

Here a *parameter choice* is an assignment of values to all the parameters of an aligner that may be varied when computing an alignment. Typically an aligner has multiple parameters whose values can be specified, such as the substitution scoring matrix and gap penalties for its alignment scoring function. We represent a parameter choice by a vector whose components assign values to all these parameters. (So for protein sequence alignment, a typical parameter choice is a 3-vector specifying the (i) substitution matrix, (ii) gap-open penalty, and (iii) gap-extension penalty.) The universe U of parameter choices specifies all the possible parameter choices that might be used for advising. A particular advisor will use a subset $P \subseteq U$ of parameter choices that it considers when advising. In the special case $|P| = 1$, the single parameter choice in set P that is available to the advisor is effectively a *default* parameter choice for the aligner.

Note that since a reference alignment R_i is known for each benchmark B_i, the true accuracy of each alternate alignment A_{ij} for benchmark B_i can be measured by comparing alignment A_{ij} to the reference R_i. Thus for a set $P \subseteq U$ of parameter choices available to an advisor, the most accurate parameter choice $j \in P$ to use on benchmark B_i can be determined in principle by comparing the resulting alternate alignments A_{ij} to R_i and picking the one of highest true accuracy. When aligning sequences in practice, a reference alignment is not known, so an advisor will instead use its estimator to pick the parameter choice $j \in P$ whose resulting alignment A_{ij} has highest *estimated* accuracy.

In the problem formulations below, this underlying training data is summarized by

- the *accuracies* a_{ij} of alternate alignments A_{ij}, where accuracy a_{ij} measures how well the computed alignment A_{ij} agrees with the reference alignment R_i, and
- the *feature vectors* F_{ij} of these alignments A_{ij}, where each vector F_{ij} lists the values for A_{ij} of the estimator's feature functions.

For an estimator that uses d feature functions, each feature vector F_{ij} is a vector of d feature values,

$$F_{ij} \;=\; (g_{ij1}\; g_{ij2}\; \cdots\; g_{ijd})\,,$$

where each feature value g_{ijh} is a real number satisfying $0 \leq g_{ijh} \leq 1$. Feature vector F_{ij} is used by the advisor to evaluate its accuracy estimator E on alignment A_{ij}. Let the *coefficients* of the estimator E be given by vector

$$c \;=\; (c_1\; c_2\; \cdots\; c_d)\,.$$

Then the value of accuracy estimator E on alignment A_{ij} is given by the inner product

$$E_c(A_{ij}) \;=\; c \cdot F_{ij} \;=\; \sum_{1 \leq h \leq d} c_h\, g_{ijh}\,. \qquad (1)$$

Informally, the objective function that the problem formulations seek to maximize is the average accuracy

achieved by the advisor across the suite of benchmarks in the training set. The benchmarks may be nonuniformly weighted in this average to correct for bias in the training data, such as the over-representation of easy benchmarks that typically occurs in standard benchmark suites.

A subtle issue that the formulations must take into account is that when an advisor is selecting a parameter choice via its estimator, there can be ties in the estimator value, so there may not be a unique parameter choice that maximizes the estimator. In this situation, we assume that the advisor *randomly* selects a parameter choice among those of maximum estimator value. Given this randomness, we measure the performance of an advisor on an input by its *expected* accuracy on that input.

Furthermore, in practice any accuracy estimator inherently has *error* (otherwise it would be equivalent to true accuracy), and a robust formulation for learning an advisor should be tolerant of error in the estimator. Let $\epsilon \geq 0$ be a given error *tolerance*, and P be the set of parameter choices used by an advisor. We define the set $\mathcal{O}_i(P)$ of parameter choices that the advisor could potentially *output* for benchmark B_i as

$$\mathcal{O}_i(P) \;=\; \Big\{ j \in P \;:\; E_c(A_{ij}) \;\geq\; e_i^* \;-\; \epsilon \Big\}, \qquad (2)$$

where $e_i^* \;:=\; \max\big\{ E_c(A_{i\tilde{j}}) \;:\; \tilde{j} \in P \big\}$ is the maximum estimator value on benchmark B_i. The parameter choice output by an advisor on benchmark B_i is selected uniformly at random among those in $\mathcal{O}_i(P)$. Note that when $\epsilon = 0$, set $\mathcal{O}_i(P)$ is simply the set of parameter choices that are tied for maximizing the estimator. A nonzero tolerance $\epsilon > 0$ can aid in learning an advisor that has improved generalization to testing data.

The *expected accuracy* achieved by the advisor on benchmark B_i using set P is then

$$\mathcal{A}_i(P) \;=\; \frac{1}{|\mathcal{O}_i(P)|} \sum_{j \,\in\, \mathcal{O}_i(P)} a_{ij}\,. \qquad (3)$$

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

Formally, we want an advisor that maximizes the following *objective function*,

$$f_c(P) \;=\; \sum_i w_i \; \mathcal{A}_i(P)\,, \qquad (4)$$

where i indexes the benchmarks, and w_i is the weight placed on benchmark B_i. (The benchmark weights are to correct for possible sampling bias in the training data.) In words, objective $f_c(P)$ is the expected accuracy of the parameter choices selected by the advisor averaged across the weighted training benchmarks, using advisor set P and the estimator given by coefficients c. We write the objective function as $f(P)$ without subscript c when the estimator coefficient vector c is fixed or understood from context.

We now define the problem of finding an *optimal set* of parameter choices for advising with a given estimator. The running time of an advisor grows with the number of parameter choices it considers, so the problem formulation bounds the allowed cardinality of the set that it finds, and seeks the best set within this cardinality bound.

In the definition, $\mathcal{Q}$ denotes the set of rational numbers.

Definition 1 (Advisor Set). The *Advisor Set* problem is the following. The input is

- cardinality bound $k \geq 1$,
- universe U of parameter choices,
- weights $w_i \in \mathcal{Q}$ on the training benchmarks B_i, where each $w_i \geq 0$ and $\sum_i w_i = 1$,
- accuracies $a_{ij} \in \mathcal{Q}$ of the alternate alignments A_{ij}, where each $0 \leq a_{ij} \leq 1$,
- feature vectors $F_{ij} \in \mathcal{Q}^d$ for the alternate alignments A_{ij}, where each feature value g_{ijh} in vector F_{ij} satisfies $0 \leq g_{ijh} \leq 1$,
- estimator coefficient vector $c \in \mathcal{Q}^d$, where in vector c each coefficient $c_h \geq 0$ and $\sum_h c_h = 1$, and
- error tolerance $\epsilon \in \mathcal{Q}$ where $\epsilon \geq 0$.

The output is

- advisor set $P \subseteq U$ of parameter choices with $|P| \leq k$,

that maximizes objective $f_c(P)$ given by equation (4).

As Section 5 later shows, Advisor Set is NP-complete, so finding an optimal solution is hard. We next show that a natural greedy approach will find a near-optimal solution.

4 Approximation Algorithm for Learning Advisor Sets

As Advisor Set is NP-complete, it is unlikely we can efficiently find advisor sets that are *optimal*; we can, however, efficiently find advisor sets that are guaranteed to be *close* to optimal, in the following sense. An *α-approximation algorithm* for a maximization problem, where $\alpha < 1$, is a polynomial-time algorithm that finds a feasible solution whose value under the objective function is at least factor α times the value of an optimal solution. Factor α is called the *approximation ratio*. In this section we show that for any constant ℓ with $\ell \leq k$, there is a simple approximation algorithm for Advisor Set that achieves approximation ratio ℓ/k.

For constant ℓ, the optimal advisor set of cardinality at most ℓ can be found in polynomial time by exhaustive search (since when ℓ is a constant there are polynomially-many subsets of size at most ℓ). The following natural approach to Advisor Set builds on this idea, by starting with an optimal advisor set of size at most ℓ, and greedily augmenting it to one of size at most k. Since augmenting an advisor set by adding a parameter choice can worsen its value under the objective function, even if augmented in the best possible way, the procedure `Greedy` given below outputs the best advisor set found across all cardinalities.

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

We now show this natural greedy procedure is an approximation algorithm for Advisor Set.

Theorem 1 (Approximation Ratio). Procedure `Greedy` is an (ℓ/k)-approximation algorithm for Advisor Set with cardinality bound k, and any constant ℓ with $\ell \leq k$.

Proof: The basic idea of the proof is to use averaging over all subsets of size ℓ from the optimal advisor set of size at most k, in order to relate the objective function value of the set found by `Greedy` to the optimal solution.

To prove the approximation ratio, let

- P^* be the optimal advisor set of size at most k,
- $\widetilde{P}$ be the optimal advisor set of size at most ℓ,
- P be the advisor set output by `Greedy`,
- $\mathcal{S}$ be the set of all subsets of P^* that have size ℓ,
- $\widetilde{k}$ be the size of P^*, and
- $\widetilde{\ell}$ be the size of $\widetilde{P}$.

Note that if $\widetilde{k} < \ell$, then the greedy advisor set P is actually optimal and the approximation ratio holds. So assume $\widetilde{k} \geq \ell$, in which case $\mathcal{S}$ is nonempty. Then

$$\begin{aligned} f(P) &\geq f(\widetilde{P}) \\ &\geq \max_{Q \in \mathcal{S}} f(Q) \qquad (5) \\ &\geq \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} f(Q) \\ &= \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} \sum_{i} w_i \, \mathcal{A}_i(Q) \\ &= \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} \sum_{i} \sum_{j \in \mathcal{O}_i(Q)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(Q)|} \\ &= \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} \sum_{j \in Q} \sum_{i \,:\, j \in \mathcal{O}_i(Q)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(Q)|}, \qquad (6) \end{aligned}$$

where inequality (5) holds because $\widetilde{P}$ is an optimal set of size at most ℓ and each Q is a set of size ℓ, while equation (6) just changes the order of summation on i and j.

Note that for any subset $Q \subseteq P^*$ and any fixed parameter choice $j \in Q$, the following relationship on sets of benchmarks holds:

$$\Big\{ i \,:\, j \in \mathcal{O}_i(P^*) \Big\} \;\subseteq\; \Big\{ i \,:\, j \in \mathcal{O}_i(Q) \Big\}, \qquad (7)$$

since if choice j is within tolerance ϵ of the highest estimator value for P^*, then j is within ϵ of the highest value for Q.

Continuing from equation (6), applying relationship (7) to index i of the innermost sum and observing that the terms lost are nonnegative, yields the following inequality (8):

$$\begin{aligned} f(P) &\geq \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} \sum_{j \in Q} \sum_{i \,:\, j \in \mathcal{O}_i(Q)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(Q)|} \\ &\geq \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} \sum_{j \in Q} \sum_{i \,:\, j \in \mathcal{O}_i(P^*)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(Q)|}. \qquad (8) \end{aligned}$$

Now define, for each benchmark i, a parameter choice $J(i)$ from P^* of highest estimator value,

$$J(i) \;\in\; \operatorname*{argmax}_{j \in P^*} \Big\{ E(A_{ij}) \Big\},$$

where ties in the maximum estimator value are broken arbitrarily. Observe that when $J(i) \in Q$, the relationship $\mathcal{O}_i(Q) \subseteq \mathcal{O}_i(P^*)$ holds, since then both Q and P^* have the same highest estimator value (and $Q \subseteq P^*$). Thus when $J(i) \in Q$,

$$\big|\mathcal{O}_i(Q)\big| \;\leq\; \big|\mathcal{O}_i(P^*)\big|. \qquad (9)$$

Returning to inequality (8), and applying relationship (9) in inequality (10) below,

$$\begin{aligned} f(P) &\geq \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} \sum_{j \in Q} \sum_{i \,:\, j \in \mathcal{O}_i(P^*)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(Q)|} \\ &= \frac{1}{|\mathcal{S}|} \sum_{i} \sum_{Q \in \mathcal{S}} \sum_{j \in \mathcal{O}_i(P^*)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(Q)|} \\ &\geq \frac{1}{|\mathcal{S}|} \sum_{i} \sum_{Q \in \mathcal{S} \,:\, J(i) \in Q} \sum_{j \in \mathcal{O}_i(P^*)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(Q)|} \\ &\geq \frac{1}{|\mathcal{S}|} \sum_{i} \sum_{Q \in \mathcal{S} \,:\, J(i) \in Q} \sum_{j \in \mathcal{O}_i(P^*)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(P^*)|} \qquad (10) \\ &= \frac{1}{|\mathcal{S}|} \sum_{i} \Big| \big\{ Q \in \mathcal{S} \,:\, J(i) \in Q \big\} \Big| \sum_{j \in \mathcal{O}_i(P^*)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(P^*)|} \\ &= \frac{\binom{\widetilde{k}-1}{\ell-1}}{\binom{\widetilde{k}}{\ell}} \sum_{i} \sum_{j \in \mathcal{O}_i(P^*)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(P^*)|} \\ &= \Big(\ell \Big/ \widetilde{k} \Big) \; f(P^*) \\ &\geq (\ell/k) \; f(P^*). \end{aligned}$$

Thus `Greedy` achieves approximation ratio at least ℓ/k.

Finally, to bound the running time of `Greedy`, consider an input instance with d features, n benchmarks, and m parameter choices in universe U. There are at most m^ℓ subsets of U of size at most ℓ, and evaluating objective function f on such a subset takes $O(d\ell n)$ time, so finding the optimal subset of size at most ℓ in the first step of `Greedy` takes $O(d\ell n m^\ell)$ time. The remaining for-loop considers at most k cardinalities, at most m parameter choices for each cardinality, and evaluates the objective function for each parameter choice on a subset of size at most k, which takes $O(dk^2mn)$ time. Thus the total time for `Greedy` is $O(d\ell n m^\ell + dk^2mn)$. For constant ℓ, this is polynomial time. □

In practice, we can compute optimal advisor sets of size up to $\ell = 5$ by exhaustive enumeration, as shown in Section 6.2. Finding an optimal advisor set of size $k = 10$, however, is currently far out of reach. Nevertheless, Theorem 1 shows we can still find reasonable approximations even for such large advisor sets, since for $\ell = 5$ and $k = 10$, `Greedy` is a $(1/2)$-approximation algorithm.

We next show it is not possible to prove a greater approximation ratio than in Theorem 1, as that ratio is in fact tight.

Theorem 2 (Tightness of Approximation Ratio). The approximation ratio ℓ/k for algorithm `Greedy` is tight.

Proof: Since the ratio is obviously tight for $\ell = k$, assume $\ell < k$. For any arbitrary constant $0 < \delta < 1 - (\ell/k)$, and for any error tolerance $0 \leq \epsilon < 1$, consider the following infinite class of instances of Advisor Set with:

- benchmarks $1, 2, \ldots, n$,
- benchmark weights $w_i = 1/n$,
- cardinality bound $k = n$, and
- universe $U = \{0, 1, \ldots, n\}$ of $n+1$ parameter choices.

The estimator values for all benchmarks i are,

$$E(A_{ij}) \;=\; \begin{cases} 1, & j = 0; \\ (1-\epsilon)/2, & i = j > 0; \\ 0, & \text{otherwise;} \end{cases}$$

which can be achieved by appropriate feature vectors F_{ij}. The alternate alignment accuracies for all benchmarks i are,

$$a_{ij} \;=\; \begin{cases} (\ell/k) + \delta, & j = 0; \\ 1, & i = j > 0; \\ 0, & \text{otherwise.} \end{cases}$$

For such an instance of Advisor Set, an optimal set of size at most k is $P^* = \{1, \ldots, n\}$, which achieves $f(P^*) = 1$. Every optimal set $\widetilde{P}$ of size at most $\ell < k$ satisfies $\widetilde{P} \supseteq \{0\}$: it cannot include all of parameter choices $1, 2, \ldots, n$, so to avoid getting accuracy 0 on a benchmark it must contain parameter choice $j = 0$. Moreover, every such set $\widetilde{P} \supseteq \{0\}$ has average accuracy $f\big(\widetilde{P}\big) = (\ell/k) + \delta$: parameter choice $j = 0$ has the maximum estimator value 1 on every benchmark, and no other parameter choice $j \neq 0$ has estimator value within ϵ of the maximum, so on every benchmark $\mathcal{A}_i\big(\widetilde{P}\big) = (\ell/k) + \delta$. Furthermore, every greedy augmentation $P \supseteq \widetilde{P}$ also has this same average accuracy $f(P) = f\big(\widetilde{P}\big)$. Thus on this instance the advisor set P output by `Greedy` has approximation ratio exactly

$$\frac{f(P)}{f(P^*)} \;=\; \frac{\ell}{k} \;+\; \delta\,.$$

Now suppose the approximation ratio from Theorem 1 is not tight, in other words, that an even better approximation ratio $\alpha > \ell/k$ holds. Then take $\delta = \big(\alpha - (\ell/k)\big)/2$, and run `Greedy` on the above input instance. On this instance, `Greedy` only achieves ratio

$$\frac{\ell}{k} \;+\; \delta \;=\; \frac{1}{2}\left(\frac{\ell}{k} \;+\; \alpha\right) \;<\; \alpha\,,$$

a contradiction. So the approximation ratio is tight. □

5 Complexity of Learning Optimal Advisors

We now prove that Advisor Set, the problem of learning an optimal parameter set for an advisor (given by Definition 1 of Section 3) is NP-complete, and hence is unlikely to be efficiently solvable in the worst-case. As is standard, we prove NP-completeness for a decision version of this optimization problem, which is a version whose output is a yes/no answer (as opposed to a solution that optimizes an objective function).

The *decision version* of Advisor Set has an additional input $\ell \in \mathcal{Q}$, which will lower bound the objective function. The decision problem is to determine, for the input instance $k, U, w_i, a_{ij}, F_{ij}, c, \epsilon, \ell$, whether or not there exists a set $P \subseteq U$ with $|P| \leq k$ for which the objective function has value $f_c(P) \geq \ell$.

Theorem 3 (NP-completeness of Advisor Set). The decision version of Advisor Set is NP-complete.

Proof: We use a reduction from the *Dominating Set* problem, which is NP-complete [17, problem GT2]. The input to Dominating Set is an undirected graph $G = (V, E)$ and an integer k, and the problem is to decide whether or not G contains a vertex subset $S \subseteq V$ with $|S| \leq k$ such that every vertex in V is in S or is adjacent to a vertex in S. Such a set S is called a *dominating set* for G.

Given an instance G, k of Dominating Set, we construct an instance $U, w_i, a_{ij}, F_{ij}, c, \epsilon, \ell$ of the decision version of Advisor Set as follows. For the cardinality bound use the same value k, for the number of benchmarks take $n = |V|$, and index the universe of parameter choices by $U = \{1, \ldots, n\}$; have only one feature ($d = 1$) with estimator coefficients $c = 1$; use weights $w_i = 1/n$, error tolerance $\epsilon = 0$, and lower bound $\ell = 1$. Let the vertices of G be indexed $V = \{1, \ldots, n\}$. (So both the set of benchmarks and the universe of parameter choices in essence correspond to the set of vertices V of graph G.) Define the *neighborhood* of vertex i in G to be $N(i) := \big\{j : (i,j) \in E\big\} \cup \{i\}$, which is the set of vertices adjacent to i, including i itself. For the alternate alignment accuracies, take $a_{ij} = 1$ when $j \in N(i)$; otherwise, $a_{ij} = 0$. For the feature vectors, assign $F_{ij} = a_{ij}$.

We claim G, k is a yes-instance of Dominating Set iff $k, U, w_i, a_{ij}, F_{ij}, c, \epsilon, \ell$ is a yes-instance of Advisor Set.

To show the forward implication, suppose G has a dominating set $S \subseteq V$ with $|S| \leq k$, and consider the advisor set $P = S$. With the above construction, for every benchmark, set $\mathcal{O}_i(P) = N(i) \cap S$, which is nonempty (since S is a dominating set for G). So $\mathcal{A}_i(P) = 1$ for all benchmarks. Thus for this advisor set P, the objective function has value $f_c(P) = 1 \geq \ell$.

For the reverse implication, suppose advisor set P achieves objective value $\ell = 1$. Since P achieves value 1, for every benchmark it must be that $\mathcal{A}_i(P) = 1$. By construction of the a_{ij}, this implies that in G every vertex $i \in V$ is in P or is adjacent to a vertex in P. Thus set $S = P$, which satisfies $|S| \leq k$, is a dominating set for G.

This reduction shows Advisor Set is NP-hard, as the instance of Advisor Set can be constructed in polynomial time. Furthermore, it is in NP, as we can nondeterministically guess an advisor set P, and then check whether its cardinality is at most k and its objective value is at least ℓ in polynomial time. Thus Advisor Set is NP-complete. □

Note that the proof of Theorem 3 shows Advisor Set is NP-complete for the special case of a *single feature*, error tolerance zero, when all accuracies and feature values are binary, and benchmarks are uniformly weighted.

In general, we would like to find an optimal *parameter advisor*, which requires simultaneously finding both the best possible parameter *set* and the best possible accuracy *estimator*. We define the general problem of constructing an optimal parameter advisor as follows.

Definition 2 (Optimal Advisor). The input to *Optimal Advisor* is cardinality bound k, parameter universe U, benchmark weights w_i, alignment accuracies a_{ij}, feature vectors F_{ij}, and error tolerance ϵ. The output is

- advisor set $P \subseteq U$ with $|P| \leq k$, and
- estimator coefficients $c \in \mathcal{Q}^d$ with $c_i \geq 0$ and $\sum_i c_i = 1$,

that maximize objective $f_c(P)$ defined in equation (4).

The *decision version* of Optimal Advisor, similar to the decision version of Advisor Set, has an additional input ℓ that lower bounds the objective function.

We next prove that Optimal Advisor is NP-complete. While its NP-hardness follows from Advisor Set, the difficulty is in proving that this more general problem is still in the class NP.

Theorem 4 (NP-completeness of Optimal Advisor). The decision version of Optimal Advisor is NP-complete.

Proof: The proof of Theorem 3 shows Advisor Set remains NP-hard for the special case of a single feature. To prove the decision version of Optimal Advisor is NP-hard, we use *restriction*: we simply reduce Advisor Set with a single feature to Optimal Advisor (reusing the instance of Advisor Set for Optimal Advisor). On this restricted input with $d = 1$, Optimal Advisor is equivalent to Advisor Set, so Optimal Advisor is also NP-hard.

We now show the general Optimal Advisor problem is in class NP. To decide whether its input is a yes-instance, after first nondeterministically guessing parameter set $P \subseteq U$ with $|P| \leq k$, we then make for each benchmark i a nondeterministic guess for its sets $\mathcal{O}_i(P)$ and $\mathcal{M}_i(P) := \operatorname{argmax}\{c \cdot F_{ij} : j \in P\}$, *without* yet knowing the coefficient vector c. Call $\widetilde{O}_i$ the guess for set $\mathcal{O}_i(P)$, and $\widetilde{M}_i$ the guess for set $\mathcal{M}_i(P)$, where $\widetilde{M}_i \subseteq \widetilde{O}_i \subseteq P$. To check whether a coefficient vector c exists that satisfies $\mathcal{O}_i(P) = \widetilde{O}_i$ and $\mathcal{M}_i(P) = \widetilde{M}_i$, we construct the following linear program with variables $c = (c_1 \cdots c_d)$ and ξ. The objective function for the linear program is to maximize the value of variable ξ. The constraints are: $c_h \geq 0$ and $\sum_{1 \leq h \leq d} c_h = 1$; $0 \leq \xi \leq 1$; for all benchmarks i and all parameter choices $j^* \in \widetilde{M}_i$ and $j \notin \widetilde{M}_i$,

$$c \cdot F_{ij^*} \geq c \cdot F_{ij} + \xi\,;$$

for all benchmarks i and all parameter choices $j, \tilde{j} \in \widetilde{M}_i$,

$$c \cdot F_{ij} = c \cdot F_{i\tilde{j}}\,;$$

for all benchmarks and all parameter choices $j^* \in \widetilde{M}_i$ and $j \in \widetilde{O}_i$,

$$c \cdot F_{ij} \geq c \cdot F_{ij^*} - \epsilon\,.$$

This linear program can be solved in polynomial time. If it has a feasible solution, then it has an optimal solution (as its objective function is bounded). In an optimal solution c^*, ξ^* we check whether $\xi^* > 0$. If this condition holds, the guessed sets $\widetilde{O}_i$, $\widetilde{M}_i$, correspond to actual sets $\mathcal{O}_i(P)$ and $\mathcal{M}_i(P)$ for an estimator. For each benchmark i, we then evaluate $\mathcal{A}_i(P)$, and check whether $\sum_i w_i \mathcal{A}_i(P) \geq \ell$. Note that after guessing the sets P, $\widetilde{O}_i$, and $\widetilde{M}_i$, the rest of the computation runs in polynomial time. Thus Optimal Advisor is in NP. □

We next turn to experimental evaluation on real data.

6 Experimental results

We evaluate the performance of our approach to learning parameter sets through experiments on a collection of protein multiple sequence alignment benchmarks. A full description of the benchmarks, and the construction of a universe U of parameter choices appropriate for protein alignment, is given in [4] and is briefly summarized below. In the experiments, we compare parameter advisors that use five different estimators from the literature: `MOS` [12], `PredSP` [10], `GUIDANCE` [14], `Facet` [4], and `TCS` [11].

The benchmark suites used in our experiments consist of reference alignments of proteins that are largely induced by structurally aligning their known three-dimensional structures. In particular, we use the `BENCH` suite of Edgar [18], supplemented by a selection from the `PALI` suite of Balaji et al. [19]. The full benchmark collection we use consists of 861 reference alignments. When evaluating the `GUIDANCE` estimator, which only applies to alignments with at least four sequences, we use a subset of this collection consisting of all 605 reference alignments with this many sequences.

As is common in benchmark suites, easy-to-align benchmarks are highly over-represented in this collection. To correct for this bias toward easy benchmarks when evaluating average advising accuracy, we binned the 861 benchmarks in our collection by *hardness*, which we measured by the true accuracy of the alignment of the benchmark's sequences computed using the multiple alignment tool `Opal` [2], [3] under its optimal default parameter choice. We then divided the full range $[0, 1]$ of accuracies into 10 bins, where bin b for $b = 1, \ldots, 10$ contains hardness interval $\big((b-1)/10,\ b/10\big]$. The weight w_i of benchmark B_i falling in bin b that we used for training is $w_i = (1/10)(1/n_b)$, where n_b is the number of benchmarks in bin b. These weights w_i are such that each bin contributes equally to objective $f(P)$, which in effect measures advising accuracy uniformly averaged across the full range of hardnesses.

Notice that with this uniform weighting of bins, the singleton advising set P containing *only* the optimal default parameter choice will tend to an average advising accuracy $f(P)$ of 50% (illustrated later in Figure 3). This establishes, as a point of reference, average accuracy 50% as the *baseline* against which to compare advising performance.

Note that if we instead measure advising accuracy by uniformly averaging over *benchmarks*, then the predominance of easy benchmarks (for which little improvement is possible over the default parameter choice) makes both good and bad advisors tend to an average accuracy of nearly 100%. By uniformly averaging over *bins*, we can discriminate among advisors, though the average advising accuracies we report are now pulled down from 100% toward 50%.

For each benchmark in our collection, we generated alternate alignments of its sequences using the `Opal` aligner invoked with each parameter choice from universe U. A *parameter choice* for `Opal` is a five-tuple $(\sigma, \gamma_I, \gamma_E, \lambda_I, \lambda_E)$ of parameter values, where

- σ specifies the amino acid substitution scoring matrix,
- γ_E and λ_E specify the gap-open and gap-extension penalties for *external* gaps in the alignment (also called terminal gaps) that insert or delete prefixes or suffixes of the sequences, and
- γ_I and λ_I specify the corresponding gap penalties for *internal* gaps (also called non-terminal gaps).

To form the universe U of parameter choices, we first considered eight substitution matrices from the `BLOSUM` [20] and `VTML` [21] families (whose entries were scaled to inte-

gers in the range $[0, 100]$), combined with over 2,100 four-tuples of gap penalties (whose values varied in a range around the default parameter values for `Opal`). This initial set of roughly 16,900 parameter choices (each substitution matrix paired with each gap-penalty assignment) was then reduced by selecting, for each of the 10 hardness bins, their 25 most accurate parameter choices: those with the highest average accuracy on the benchmarks in the bin (when used in `Opal`). Unioning these top choices from all bins (and removing duplicates) gave the final universe U, which consists of 243 parameter choices.

To generate training and testing sets for our experiments on learning advisor sets, we used 12-*fold cross validation*. For each hardness bin, we evenly and randomly partitioned the benchmarks in the bin into twelve groups; we then formed twelve splits of the entire collection of benchmarks into a training class and a testing class, where each split placed one group in a bin into the testing class and the other eleven groups in the bin into the training class; finally, for each split we generated a *training set* and a *testing set* of example alignments as follows: for each benchmark B in a training or testing class, we generated $|U|$ example alignments in the respective training or testing set by running `Opal` on B with each parameter choice from U. An estimator learned on the examples from a training set was evaluated on examples from the corresponding testing set. The results we report are averages over twelve folds, where each *fold* is one of these pairs of associated training and testing sets. (Note that across the twelve folds, every example is tested on exactly once.) Each fold contains over 190,000 training examples.

When evaluating the `GUIDANCE` estimator, we used 4-fold cross validation on the reduced benchmark collection described earlier, with folds generated by the above procedure. Each of these folds has over 109,000 training examples.

6.1 The `Facet` estimator

The quality of an advisor strongly depends on the quality of its *estimator*. In turn, the quality of the estimator constructed by the approach of Section 2.1, embodied in `Facet` [5], [7], strongly depends on the quality of its *feature functions*. These features should correlate well with true accuracy, be efficiently computable, and have bounded value. In practice, the best features for accuracy estimation of protein sequence alignments tend to use secondary structure predicted from the sequences; in our experiments, we predicted protein secondary structure using `PSIPRED` [22].

Our accuracy estimator `Facet` is a linear combination of the following five feature functions, which we summarize briefly. Full details are in [4].

(1) The feature *Secondary Structure Blockiness* measures the fraction of substitutions in the alignment that are in an optimal packing of secondary structure blocks, where a *secondary structure block* is a subset of the alignment's rows and a consecutive interval of its columns such that all amino acids in the block have the same predicted secondary structure.

(2) *Secondary Structure Agreement* is the probability that the amino acids paired by substitutions share the same secondary structure, based on predicted secondary structure for the surrounding sequence.

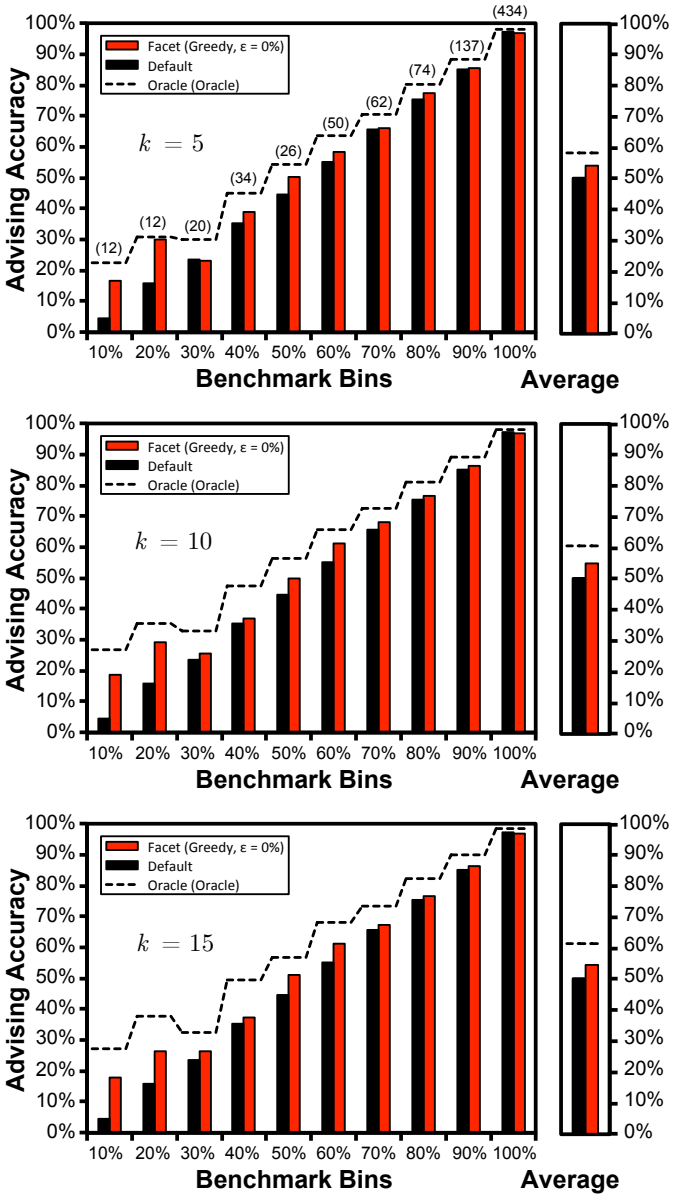

Fig. 3. *Advising accuracy of* `Facet` *within benchmark bins.* These bar charts show the advising accuracy of various approaches to finding advisor sets, for cardinality $k = 5, 10, 15$. For each cardinality, the horizontal axis of the chart on the left corresponds to benchmark bins, and the vertical bars show advising accuracy averaged over the benchmarks in each bin. Black bars give the accuracy of the optimal *default* parameter choice, and red bars give the accuracy of advising with `Facet` using the *greedy* set. The dashed line shows the limiting performance of a perfect advisor: an oracle with true accuracy as its estimator using an optimal *oracle* set. In the top chart, the numbers in parentheses above the bars are the number of benchmarks in each bin. The bar charts on the right show advising accuracy uniformly averaged over the bins.

(3) *Secondary Structure Identity* measures the fraction of substitutions whose amino acids share the same predicted secondary structure.

(4) *Gap Open Density* counts the number of runs of null characters (or dashes) in the rows of the alignment, normalized by the total length of the runs.

(5) *Average Substitution Score* is the average score of the substitutions in the alignment under the `BLOSUM62` [20] scoring matrix.

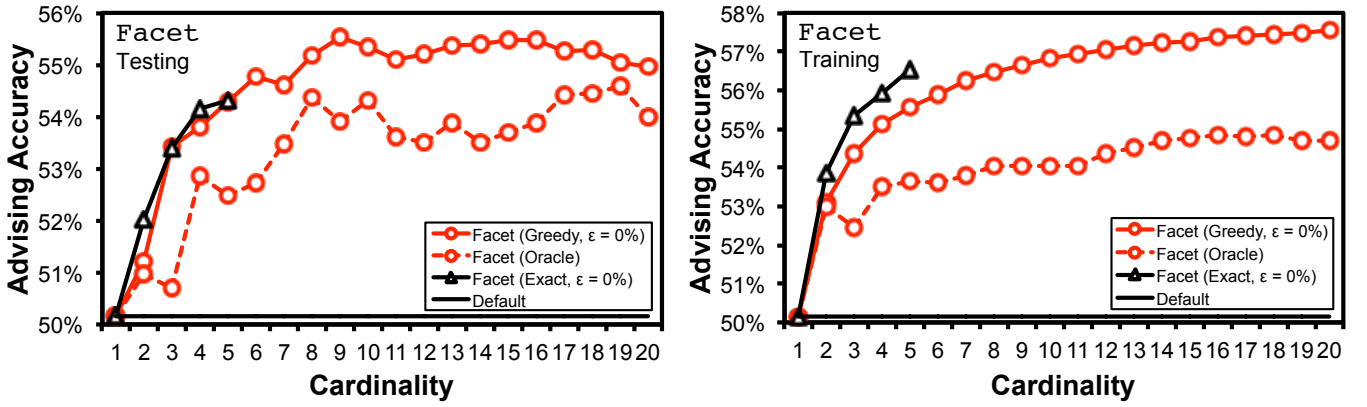

Fig. 4. *Advising using exact, greedy, and oracle sets with* `Facet`. The plots show advising accuracy using the `Facet` estimator with parameter sets learned by the optimal *exact* algorithm and the *greedy* approximation algorithm for Advisor Set, and with *oracle* sets. The horizontal axis is the cardinality of the advisor set, while the vertical axis is the advising accuracy averaged over the benchmarks. Exact sets are known only for cardinalities $k \leq 5$; greedy sets are augmented from the exact set of cardinality $\ell = 1$. The left and right plots show accuracy on the testing and training data, respectively, where accuracies are averaged over all testing or training folds.

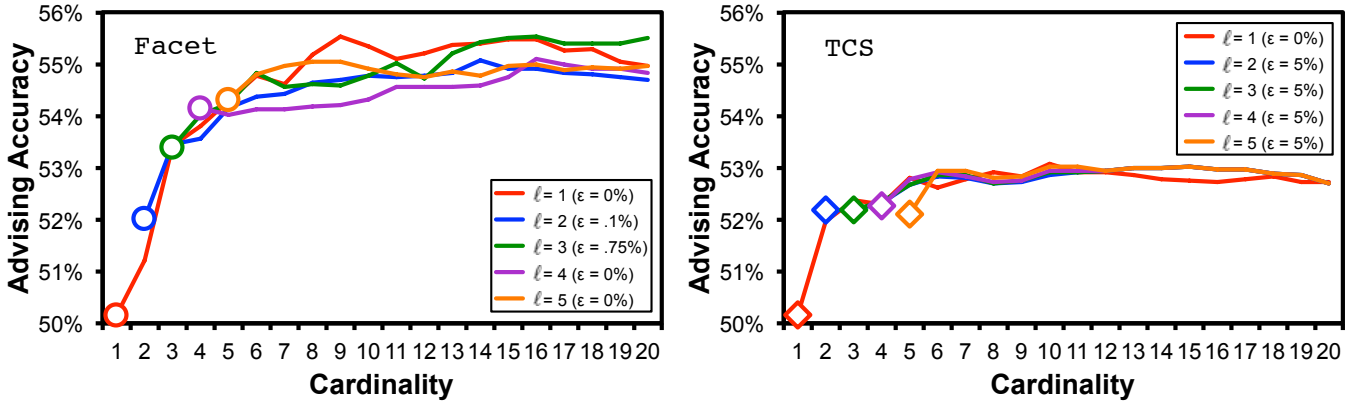

Fig. 5. *Greedily augmenting exact advisor sets.* The left and right plots show advising accuracy using the `Facet` and `TCS` estimators respectively, with advisor sets learned by procedure `Greedy`, which augments an exact set of cardinality ℓ to form a larger set of cardinality $k > \ell$. Each curve is greedily augmenting from a different exact cardinality ℓ. The horizontal axis is the cardinality k of the augmented set; the vertical axis is advising accuracy on testing data, averaged over all benchmarks and all folds.

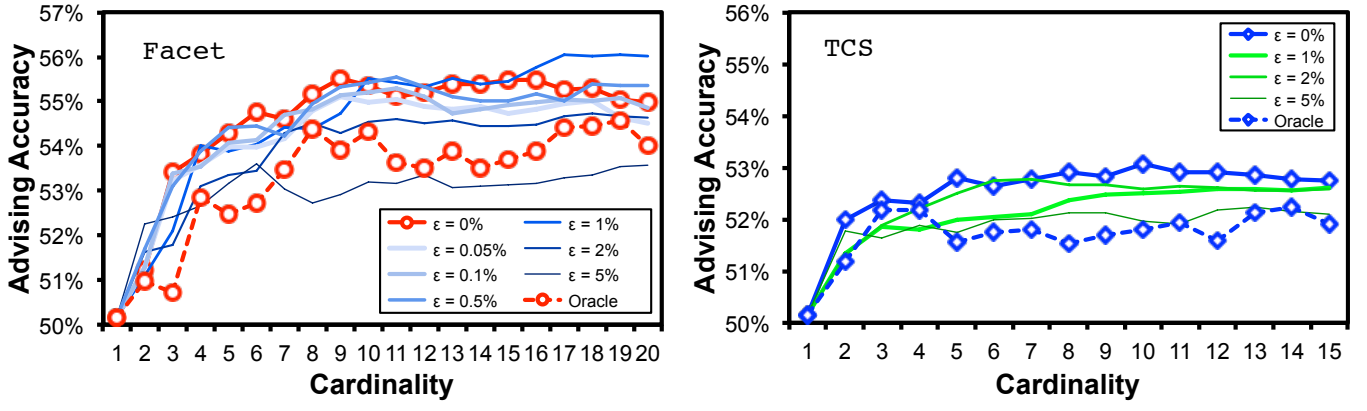

Fig. 6. *Effect of error tolerance on advising accuracy using greedy sets.* The plots show advising accuracy on testing data using greedy sets learned for the two best estimators, `Facet` and `TCS`, at various error tolerances $\epsilon \geq 0$. The plots on the left and right are for `Facet` and `TCS`, respectively. For comparison, both plots also include a curve showing performance using the estimator on oracle sets, drawn with a dashed line. The solid curves with circles and diamonds highlight the best overall error tolerance of $\epsilon = 0$.

6.2 Learning advisor sets by different approaches

We first study the advising accuracy of parameter sets learned for the Facet estimator by different approaches. Our protocol began by constructing an optimal *oracle* set for cardinalities $1 \leq k \leq 20$ for each training instance. A coefficient vector for the advisor's estimator was then found for each of these oracle sets by the difference-fitting method described in [4]. Using this estimator learned for the training data, exhaustive search was done to find optimal *exact* advisor sets for cardinalities $k \leq 5$. The optimal exact set of size $\ell = 1$ (the best default parameter choice) was then used as the starting point to find near-optimal *greedy* advisor sets by our approximation algorithm for $k \leq 20$. Each of these advisors (an advising set combined with the estimator) was then used for parameter advising in Opal, returning the computed alignment with highest estimator value. These set-finding approaches are compared based on the accuracy of the alignment chosen by the advisor, averaged across bins.

Figure 4 shows the performance of these advisor sets under twelve-fold cross validation. The left plot shows advising accuracy on the testing data averaged over the folds, while the right plot shows this on the training data.

Notice that while there is a drop in accuracy when an advising set learned using the greedy and exact methods is applied to the testing data, the drop in accuracy is greatest for the exact sets. The value of ϵ shown in the plot maximizes the accuracy of the resulting advisor on the testing data. Notice also that for cardinality $k \leq 5$ (for which exact sets could be computed), on the testing data the greedy sets are often performing as well as the optimal exact sets.

Figure 3 shows the performance within each benchmark bin when advising with Facet using greedy sets of cardinality $k = 5, 10, 15$, from top to bottom. Notice that for many bins, the performance is close to the best-possible accuracy attainable by any advisor, shown by the dashed line for a perfect oracle advisor. The greatest boost over the default parameter choice is achieved on the bottom bins that contain the hardest benchmarks.

6.3 Varying the exact set for the greedy algorithm

To find the appropriate cardinality ℓ of the initial exact solution that is augmented within approximation algorithm Greedy, we examined the advising accuracy of the greedy sets learned when using cardinalities $1 \leq \ell \leq 5$. Figure 5 shows the accuracy of the resulting advisor using greedy sets of cardinality $1 \leq k \leq 20$, augmented from exact sets of cardinality $1 \leq \ell \leq 5$, using for the estimator both Facet and TCS. (These are the two best estimators, as discussed in Section 6.5 below). The points plotted with circles show the accuracy of the optimal exact set that is used within procedure Greedy for augmentation.

Notice that the initial exact set size ℓ has relatively little effect on the accuracy of the resulting advisor; at most cardinalities, starting from the single best parameter choice ($\ell = 1$) has highest advising accuracy. This is likely due to the behavior observed earlier in Figure 4, namely that exact sets do not generalize as well as greedy sets.

6.4 Varying the error tolerance for the greedy algorithm

When showing experimental results, an error tolerance ϵ has always been used that yields the most accurate advisor on the testing data. Prior to conducting these experiments, our expectation was that a nonzero error tolerance $\epsilon > 0$ would boost the generalization of advisor sets. Figure 6 shows the effect of different values of ϵ on the testing accuracy of an advisor using greedy sets learned for the Facet and TCS estimators. (While the same values of ϵ were tried for both estimators, raw TCS scores are integers in the range $[0, 100]$ which were scaled to real values in the range $[0, 1]$, so for TCS any $\epsilon < 0.1$ is equivalent to $\epsilon = 0$.) No clear relationship between testing accuracy and error tolerance is evident, though for Facet and TCS alike, setting $\epsilon = 0$ generally gives the best overall advising accuracy.

6.5 Learning advisor sets for different estimators

In addition to learning advisor sets for Facet [4], we also learned sets for the best accuracy estimators from the literature: namely, TCS [11], MOS [12], PredSP [10], and GUIDANCE [14]. The scoring-function-based accuracy estimators TCS, PredSP, and GUIDANCE do not have any dependence on the advisor set cardinality or the training benchmarks used. The support-based estimator MOS, however, requires a set of alternate alignments in order to compute its estimator value on an alignment. In each experiment, an alignment's MOS value was computed using alternate alignments generated by aligning under the parameter choices in the oracle set; if the parameter choice being tested on was in the oracle set, it was removed from this collection of alternate alignments.

After computing the values of these estimators, exhaustive search was used to find optimal exact sets of cardinality $\ell \leq 5$ for each estimator, as well as greedy sets of cardinality $k \leq 20$ (augmenting from the exact set for $\ell = 1$).

The tendency of exact advisor sets to not generalize well is even more pronounced when accuracy estimators other than Facet are used. Figure 7 shows the performance on testing and training data of greedy, exact, and oracle advisor sets learned for the best three other estimators: TCS, MOS, and PredSP. The results for greedy advisor sets for TCS at cardinalities larger than 5 have similar trend to those seen for Facet (with now a roughly 1% accuracy improvement over the oracle set), but surprisingly with TCS its exact set always has lower testing accuracy than its greedy set. Interestingly, for MOS its exact set rarely has better advising accuracy than the oracle set. For PredSP, at most cardinalities (with the exception of $k = 3$) the exact set has higher accuracy than the greedy set on testing data, though this is offset by the low accuracy of the estimator.

We also tested GUIDANCE, Facet, and TCS on the reduced suite of all benchmarks with at least four sequences (as required by GUIDANCE). Figure 8 shows the advising accuracy of set-finding methods using these estimators on these benchmarks. Notice that on this reduced suite the results generally stay the same, though for Facet there is more of a drop in performance of the exact set from training to testing, and the set found by Greedy generally has greater accuracy on the reduced suite than the full suite.

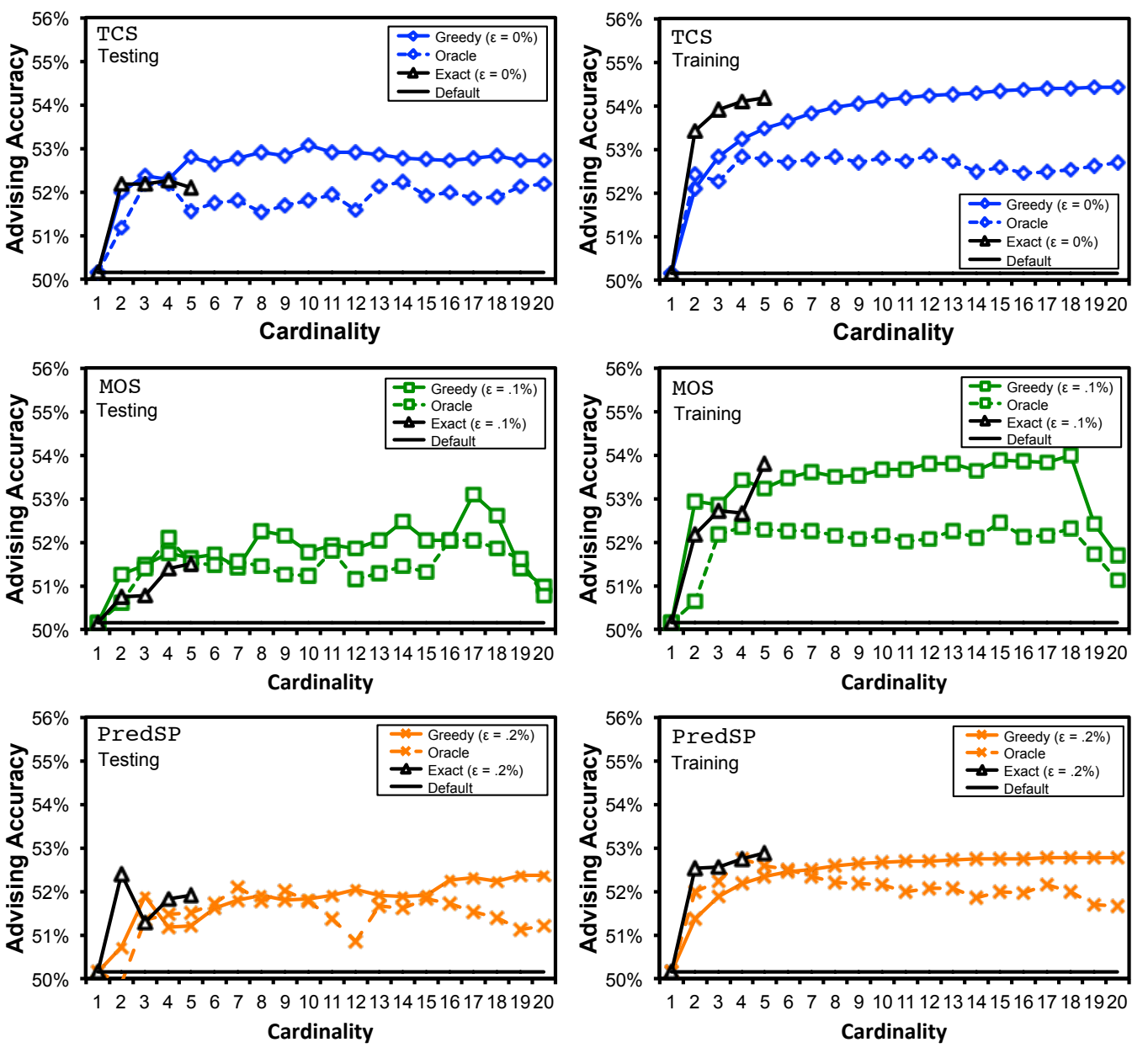

Fig. 7. *Comparing testing and training accuracies of various estimators.* The plots show the advising accuracies on testing and training data using `TCS`, `MOS`, and `PredSP` with parameter sets learned for these estimators by the *exact* and *greedy* algorithms for Advisor Set, and with *oracle* sets. From top to bottom, the estimators used are `TCS`, `MOS`, and `PredSP`, with *testing* data plotted on the left, and *training* data on the right.

Finally, a complete comparison of the advising performance of *all estimators* using greedy sets is shown in Figure 9. (The plot on the right shows advising accuracy on testing data for `GUIDANCE`, `Facet`, and `TCS` on the reduced suite of benchmarks with at least four sequences.) Advising with each of these estimators tends to eventually reach an accuracy plateau, though their performance is always boosted by using advisor sets larger than a singleton default choice. The plateau for `Facet` (the top curve in the plots) generally occurs at the greatest cardinality and accuracy.

6.6 Parameter choices in greedy advisor sets

Table 1 lists the parameter choices in the advisor sets found by the *greedy* approximation algorithm (augmenting from the optimal set of cardinality $\ell = 1$) for the `Opal` aligner with the `Facet` estimator for cardinalities $k \leq 20$, on one fold of training data. (The greedy sets vary slightly across folds.) In the table, the greedy set of cardinality k contains the parameter choices at rows 1 through k. (The entry at row 1 is the optimal *default parameter choice*.) Again a parameter choice is five-tuple $(\sigma, \gamma_I, \gamma_E, \lambda_I, \lambda_E)$, where γ_I and γ_E are gap-open penalties for non-terminal and terminal gaps respectively, and λ_I and λ_E are corresponding gap-extension penalties. The scores in the substitution matrix σ are dissimilarity values scaled to integers in the range $[0, 100]$. (The associated gap penalty values in a parameter choice relate to this range.) The accuracy column gives the average advising accuracy (in `Opal` using `Facet`) of the greedy set of cardinality k on *training* data, uniformly averaged over

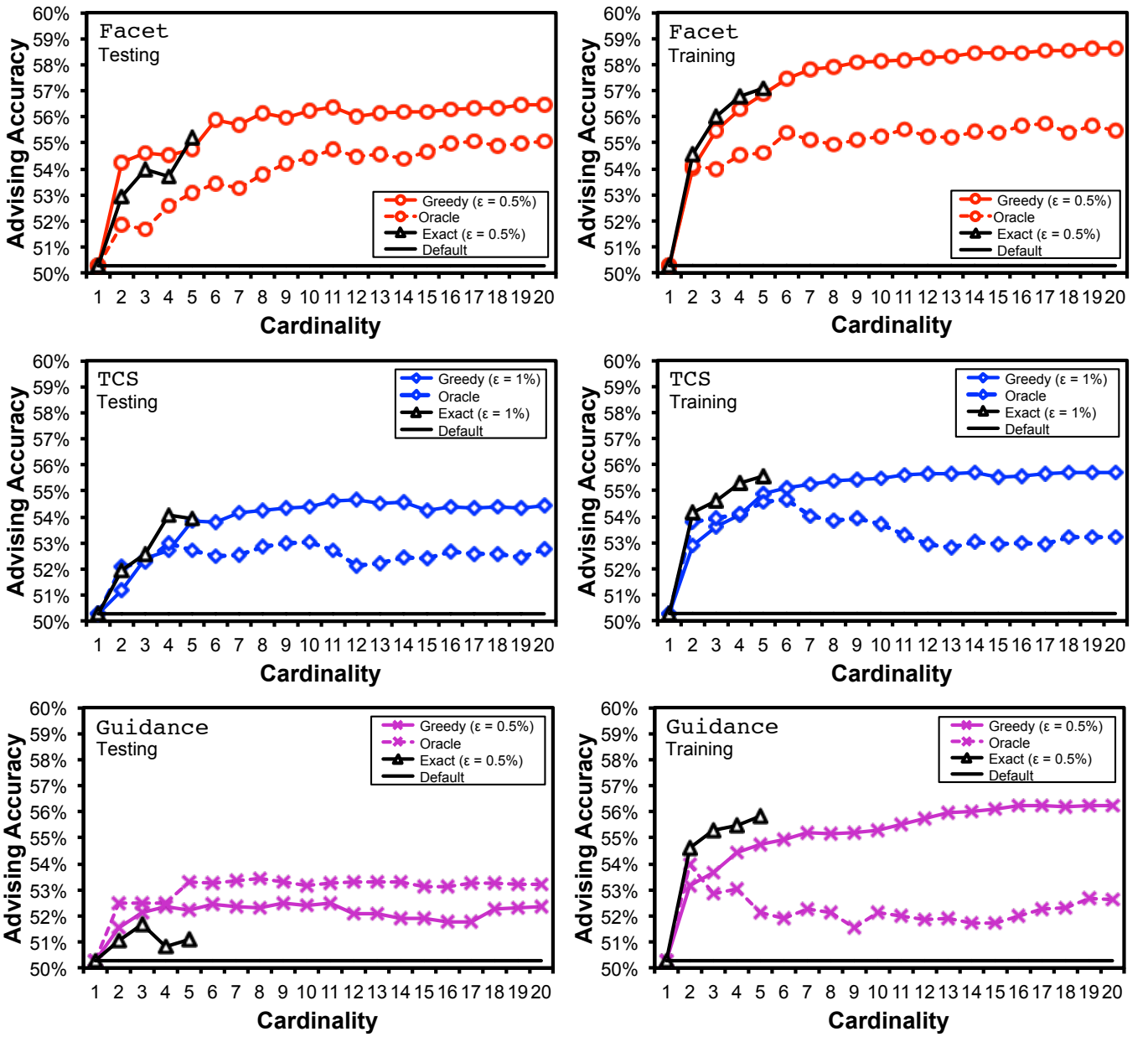

Fig. 8. *Comparing testing and training accuracies of estimators on benchmarks with at least four sequences.* The plots show advising accuracies for *testing* and *training* data on benchmarks with at least four sequences, using `Facet`, `TCS`, and `GUIDANCE` with *exact*, *greedy*, and *oracle* sets.

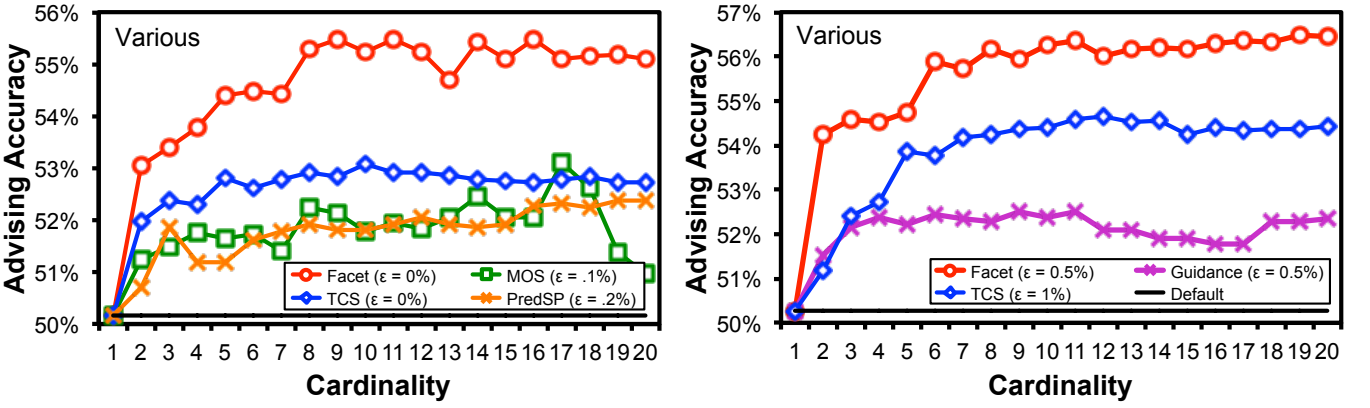

Fig. 9. *Comparing all estimators on greedy advisor sets.* The plots show advising accuracy on *greedy* sets learned for the estimators `Facet`, `TCS`, `MOS`, `PredSP`, and `GUIDANCE`. The vertical axis is advising accuracy on *testing* data, averaged over all benchmarks and all folds. The horizontal axis is the cardinality k of the greedy advisor set. Greedy sets are augmented from the exact set of cardinality $\ell = 1$. The plot on the left uses the *full* benchmark suite; the plot on the right, which includes `GUIDANCE`, uses a *reduced* suite of all benchmarks with at least four sequences.

TABLE 1
Greedy Advisor Sets for `Opal` *Using* `Facet`

Cardinality k	Parameter choice $(\sigma, \gamma_I, \gamma_E, \lambda_I, \lambda_E)$	Average advising accuracy
1	(VTML200, 50, 17, 41, 40)	51.2%
2	(VTML200, 55, 30, 45, 42)	53.4%
3	(BLSUM80, 60, 26, 43, 43)	54.5%
4	(VTML200, 60, 15, 41, 40)	55.2%
5	(VTML200, 55, 30, 41, 40)	55.6%
6	(BLSUM45, 65, 3, 44, 43)	56.1%
7	(VTML120, 50, 12, 42, 39)	56.3%
8	(BLSUM45, 65, 35, 44, 44)	56.5%
9	(VTML200, 45, 6, 41, 40)	56.6%
10	(VTML120, 55, 8, 40, 37)	56.7%
11	(BLSUM62, 80, 51, 43, 43)	56.8%
12	(VTML120, 50, 2, 45, 44)	56.9%
13	(VTML200, 45, 6, 40, 40)	57.0%
14	(VTML40, 50, 2, 40, 40)	57.1%
15	(VTML200, 50, 12, 43, 40)	57.2%
16	(VTML200, 45, 11, 42, 40)	57.3%
17	(VTML120, 60, 9, 40, 39)	57.3%
18	(VTML40, 50, 17, 40, 38)	57.4%
19	(BLSUM80, 70, 17, 42, 41)	57.4%
20	(BLSUM80, 60, 3, 42, 42)	57.6%

benchmark bins. Recall this averaging will tend to yield accuracies close to 50%.

Interestingly, while `BLOSUM62` [20] is the substitution scoring matrix most commonly used by standard aligners, it does not appear in a greedy set until cardinality $k = 11$. The `VTML` family [21] appears more often than `BLOSUM`. The plateau in advising accuracy seen in earlier plots is also indicated in this training instance, though ever more gradual improvement remains as cardinality k increases.

6.7 Shared structure across advisor sets

To assess the similarity of advisor sets found by the three approaches considered in our experiments — *greedy sets* via the approximation algorithm, *exact sets* via exhaustive search, and *oracle sets* via integer linear programming — we examine their overlap both within and between folds.

Table 2 shows the composition of the greedy, exact, and oracle sets for the training instance in one fold, at cardinality $k = 2, 3, 4$ and tolerance $\epsilon = 0$. A non-blank entry in the table indicates that the parameter choice at its row is contained in the advisor set at its column. (The column labeled "default" indicates the optimal *default parameter choice* for the fold, or equivalently, the exact set of cardinality $k = 1$.) The value in parentheses at an entry is the number of folds (for twelve-fold cross-validation) where that parameter choice appears in that advisor set. (For example, at cardinality $k = 4$, the second parameter choice (VTML200, 55, 30, 45, 42) is in the greedy, exact, and oracle sets for this particular fold, and overall is in exact sets for 9 of 12 folds, including this fold.) Surprisingly, the default parameter choice (the best single choice) never appears in the exact or oracle sets for this fold at any of the cardinalities beyond $k = 1$, and also is reused as the default in only one other fold. In general there

TABLE 2
Composition of Advisor Sets at Different Cardinalities k

Parameter choice $(\sigma, \gamma_I, \gamma_E, \lambda_I, \lambda_E)$	Advisor set: Default	Greedy	Exact	Oracle
$k = 2$				
(VTML200, 50, 17, 41, 40)	(2)	(2)		
(VTML200, 55, 30, 45, 42)		(2)	(3)	(1)
(BLSUM80, 60, 9, 43, 42)			(2)	
(BLSUM45, 65, 35, 44, 44)				(3)
$k = 3$				
(VTML200, 50, 17, 41, 40)	(2)	(2)		
(VTML200, 55, 30, 45, 42)		(3)	(5)	(1)
(BLSUM80, 60, 26, 43, 43)		(2)	(2)	
(VTML200, 55, 30, 41, 40)			(6)	
(VTML40, 45, 29, 40, 39)				(7)
(BLSUM62, 65, 16, 44, 42)				(8)
$k = 4$				
(VTML200, 50, 17, 41, 40)	(2)	(2)		
(VTML200, 55, 30, 45, 42)		(3)	(9)	(6)
(BLSUM80, 60, 26, 43, 43)		(2)		
(VTML200, 60, 15, 41, 40)		(1)		
(VTML200, 45, 6, 40, 40)			(8)	(1)
(VTML200, 55, 30, 41, 40)			(8)	
(BLSUM80, 55, 19, 43, 42)			(1)	
(VTML40, 45, 29, 40, 39)				(4)
(BLSUM62, 65, 35, 44, 42)				(3)

TABLE 3
Number of Folds Where Greedy and Exact Sets Share Parameters

Intersection cardinality	Advisor set cardinality $k = 2$	$k = 3$	$k = 4$	$k = 5$
0	9	4	3	2
1	3	5	6	5
2	0	3	3	4
3		0	0	1
4			0	0
5				0

is relatively little overlap between these advisor sets: often just one and at most two parameter choices are shared.

Table 3 examines whether this trend continues at other folds, by counting how many training instances (out of the twelve folds) share a specified number of parameter choices between their *greedy* and *exact* sets, for a given advisor set cardinality k. (For example, at cardinality $k = 4$, six training instances share exactly one parameter choice between their greedy and exact sets; in fact, the fold shown in Table 2 is one such instance.) On the whole, the two "estimator-aware" advisor sets — the greedy and exact sets — are relatively dissimilar, and never share more than $\lceil k/2 \rceil$ parameter choices.

7 Conclusion

We have introduced the new problem of learning optimal parameter sets for an advisor, and have shown that while

this problem is NP-complete, an efficient greedy approximation algorithm for learning parameter sets is remarkably close to optimal in practice. Moreover, these parameter sets significantly boost the accuracy of an aligner compared to a single default parameter choice, when advising using the best accuracy estimators from the literature.

7.1 Further research

The main frontier to next explore for further improving parameter advisors is whether new, easily-computable *feature functions* on multiple alignments can be discovered that have stronger correlation with true accuracy. Improving the accuracy estimator through better feature functions is likely to give the greatest boost in advising accuracy. Finally, it may be worth noting that the advising framework presented here is actually *independent* of multiple sequence alignment, and might be fruitfully applied *beyond alignment* to parameter advising problems in other contexts as well.

8 Acknowledgements

We would like to thank William Pearson for pointing us to the VTML family [21] of substitution scoring matrices. This work was supported by US National Science Foundation 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.

References

[1] D. DeBlasio and J. Kececioglu, "Learning parameter sets for alignment advising," *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB)*, pp. 230–239, 2014.

[2] T. J. Wheeler and J. D. Kececioglu, "Multiple alignment by aligning alignments," *Proceedings of the 15th ISCB Conference on Intelligent Systems for Molecular Biology (ISMB), Bioinformatics*, vol. 23, no. 13, pp. i559–i568, Jul. 2007.

[3] ——, "Opal: software for aligning multiple biological sequences (version 2.1.0)," http://opal.cs.arizona.edu, 2012.

[4] J. Kececioglu and D. DeBlasio, "Accuracy estimation and parameter advising for protein multiple sequence alignment," *Journal of Computational Biology*, vol. 20, no. 4, pp. 259–279, Apr. 2013.

[5] D. F. DeBlasio and J. D. Kececioglu, "Facet: software for accuracy estimation of protein multiple sequence alignments (version 1.1)," http://facet.cs.arizona.edu, 2014.

[6] I. Van Walle, I. Lasters, and L. Wyns, "SABmark: a benchmark for sequence alignment that covers the entire known fold space," *Bioinformatics*, vol. 21, no. 7, pp. 1267–1268, Mar. 2005.

[7] D. F. DeBlasio, T. J. Wheeler, and J. D. Kececioglu, "Estimating the accuracy of multiple alignments and its use in parameter advising," *Proceedings of the 16th Conference on Research in Computational Molecular Biology (RECOMB)*, pp. 45–59, 2012.

[8] C. Notredame, L. Holm, and D. G. Higgins, "COFFEE: an objective function for multiple sequence alignments," *Bioinformatics*, vol. 14, no. 5, pp. 407–422, Jun. 1998.

[9] J. D. Thompson, F. Plewniak, R. Ripp, J.-C. Thierry, and O. Poch, "Towards a reliable objective function for multiple sequence alignments," *Journal of Molecular Biology*, vol. 314, no. 4, pp. 937–951, Dec. 2001.

[10] V. Ahola, T. Aittokallio, M. Vihinen, and E. Uusipaikka, "Model-based prediction of sequence alignment quality," *Bioinformatics*, vol. 24, no. 19, pp. 2165–2171, Sep. 2008.

[11] J. M. Chang, P. D. Tommaso, and C. Notredame, "TCS: a new multiple sequence alignment reliability measure to estimate alignment accuracy and improve phylogenetic tree reconstruction," *Molecular Biology and Evolution*, vol. 31, no. 6, pp. 1625–1637, Apr. 2014.

[12] T. Lassmann and E. L. L. Sonnhammer, "Automatic assessment of alignment quality," *Nucleic Acids Research*, vol. 33, no. 22, pp. 7120–7128, Dec. 2005.

[13] G. Landan and D. Graur, "Heads or tails: a simple reliability check for multiple sequence alignments," *Molecular Biology and Evolution*, vol. 24, no. 6, pp. 1380–1383, 2007.

[14] O. Penn, E. Privman, G. Landan, D. Graur, and T. Pupko, "An alignment confidence score capturing robustness to guide tree uncertainty," *Molecular Biology and Evolution*, vol. 27, no. 8, pp. 1759–1767, Jul. 2010.

[15] J. Kim and J. Ma, "PSAR: measuring multiple sequence alignment reliability by probabilistic sampling," *Nucleic Acids Research*, vol. 39, no. 15, pp. 6359–6368, Aug. 2011.

[16] A. Bahr, J. D. Thompson, J. C. Thierry, and O. Poch, "BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations," *Nucleic Acids Research*, vol. 29, no. 1, pp. 323–326, Jan. 2001.

[17] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. New York: W.H. Freeman and Company, 1979.

[18] R. C. Edgar, "BENCH," http://www.drive5.com/bench, 2009.

[19] S. Balaji, S. Sujatha, S. S. C. Kumar, and N. Srinivasan, "PALI: a database of Phylogeny and ALIgnment of homologous protein structures," *Nucleic Acids Research*, vol. 29, no. 1, pp. 61–65, Jan. 2001.

[20] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proceedings of the National Academy of Sciences USA*, vol. 89, no. 22, pp. 10 915–10 919, Nov. 1992.

[21] T. Müller, R. Spang, and M. Vingron, "Estimating amino acid substitution models: a comparison of Dayhoff's estimator, the resolvent approach and a maximum likelihood method," *Molecular Biology and Evolution*, vol. 19, no. 1, pp. 8–13, Jan. 2002.

[22] D. T. Jones, "Protein secondary structure prediction based on position-specific scoring matrices," *Journal of Molecular Biology*, vol. 292, no. 2, pp. 195–202, Sep. 1999.

Dan DeBlasio is a PhD candidate in the Department of Computer Science at the University of Arizona. His research is in algorithms for computational biology, specifically, improving the quality of multiple sequence alignment through advising. He received his BS and MS in Computer Science from the University of Central Florida in 2007 and 2009, where his thesis work focused on reducing the memory consumption of preforming multiple sequence alignment of RNA guided by known secondary structure. Dan was a fellow of the NSF IGERT in Comparative Genomics from 2010-2013.

John Kececioglu received the PhD in Computer Science from the University of Arizona in 1991. He did postdoctoral study at the Université de Montréal and the University of California at Davis, then taught at the University of Georgia, before joining the University of Arizona in 2000 where he is an Associate Professor of Computer Science. His research is in the design, analysis, and implementation of algorithms for bioinformatics and computational biology. John is the recipient of a US National Science Foundation CAREER Award, has served on the Scientific Advisory Board of the Max Planck Institute for Computer Science, serves on the Editorial Board of *Algorithms for Molecular Biology*, and is an Associate Editor of IEEE/ACM *Transactions on Computational Biology and Bioinformatics*.