Learning Parameter Sets for Alignment Advising

Dan DeBlasio*
Department of Computer Science
The University of Arizona
Tucson AZ 85721, USA
deblasio@cs.arizona.edu

John Kececioglu
Department of Computer Science
The University of Arizona
Tucson AZ 85721, USA
kece@cs.arizona.edu

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.

Categories and Subject Descriptors

J.3 [**Life and Medical Sciences**]: Biology and genetics; F.2.2 [**Nonnumerical Algorithms and Problems**]: Computations on discrete structures

*Corresponding author

BCB'14, September 20–23, 2014, Newport Beach, CA, USA.

http://dx.doi.org/10.1145/2649387.2649448.

General Terms

Algorithms, Experimentation, Performance

Keywords

Multiple sequence 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 the parameter values when computing an alignment of their sequences, 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 [19, 20] 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.

```
d1gvoa   203 ... gsvenrarlvlevvdavcnewsad-RIGIRVSPigtfqnvdngpnee--adalyl--- ... 255
d2dora   141 ... ydfeatekllke-----vftfftk-PLGVKLPPyf---------------dlvhfdim ... 178
d1oyb    215 ... gsienrarftlevvdalveaighe-KVGLRLSPygvfnsmsggaetgivaqyayvage ... 272
d1o94a1  193 ... gslenrarfwletlekvkhavgsdcAIATRFGV----------------dtvygpqq ... 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

Figure 1: *Parameter choice affects the accuracy of computed alignments.* (a) Part of an alignment of benchmark `sup_155` from the `SABRE` [18] suite computed by `Opal` [19] using non-default parameter choice (`VTML200`, 45, 6, 40, 40); this alignment has accuracy value 75.8%, and `Facet` [10] estimator value 0.423. (b) Alignment of the same benchmark by `Opal` using the *optimal* default parameter choice (`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.

In the example of Figure 1, under the `Facet` estimator [10, 4], 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 [19] as an often-overlooked stage in multiple sequence alignment, and was first studied in depth in DeBlasio, Wheeler, and Kececioglu [5], both with regard to constructing accuracy estimators, and finding parameter sets for a perfect advisor called an oracle.

Kececioglu and DeBlasio [10] give a broad survey of accuracy estimators from the literature. Briefly, estimators can be categorized as *scoring-function-based* [15, 17, 1, 5, 3], which combine local attributes of an alignment into a score, and *support-based* [13, 12, 16, 11], 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. [5]), `TCS` (Chang et al. [3], which supersedes `COFFEE` [15]), `MOS` (Lassmann and Sonnhammer [13]), `PredSP` (Ahola et al. [1]), and `GUIDANCE` (Penn et al. [16]). Kececioglu and DeBlasio [10] 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.

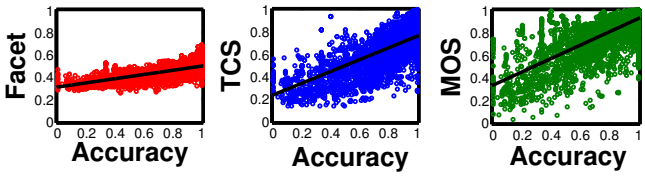


Figure 2: *Correlation of estimators with true accuracy.* Each point in a scatterplot corresponds to an alignment whose true accuracy is on the horizontal axis, and whose value under a given estimator is on the vertical axis. The scatterplots show the same set of over 4,000 alignments under the accuracy estimators `Facet` [10], `TCS` [3], and `MOS` [13].

The emphasis of our prior work [5, 10] is mainly on *accuracy estimation* for parameter advising, resulting in the `Facet` estimator [4]. 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

Section 2 next reviews the concepts of accuracy estimators and parameter 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 alignment 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 our approach to estimating the unknown accuracy of an alignment, we assume we have a collection of t real-valued feature functions $g_1(A), \ldots, g_t(A)$ on alignments A, where these functions g_i are positively correlated with true accuracy. The alignment accuracy estimators $E(A)$ we consider are linear combinations of these functions, $\sum_{1 \le i \le t} c_i\, g_i(A)$, where the coefficents $c_1, \ldots, c_t$ specify the estimator $E(A)$. Alignment *accuracies* are usually measured as real values in the range $[0, 1]$, such as the so-called *Q-score*, which is the fraction of substitutions in the ground-truth alignment that are recovered by a computed alignment A. We assume the feature functions have range $[0, 1]$, so when the coefficients form a convex combination, the resulting estimator $E(A)$ will also have range $[0, 1]$. Our prior work [5, 10] showed that this class of linear estimators is as general as polynomial estimators, as 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 $E(A)$ 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 [5, 10] 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 parameter choice p, and then "selects" the parameter choice p^* that has maximum estimated accuracy $E(A_{p^*})$. 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 [5, 10] 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 only has an accuracy estimator), and always selects the parameter choice from P that has highest true accuracy. In contrast to finding such an oracle set, 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*, averaged over a training set of alignment benchmarks.

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. We assume throughout that 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 the 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 t feature functions, each feature vector F_{ij} is a vector of t feature values,

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

where each feature value g_{ijh} is a real number satisfying $0 \le g_{ijh} \le 1$. 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

$$c \;=\; (c_1\; c_2\; \cdots\; c_t).$$

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

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

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 $\epsilon \ge 0$ 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

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

where $e_i^* := \max\big\{E_c(A_{i\tilde{\jmath}}) : \tilde{\jmath} \in P\big\}$ 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 the advisor's 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 definitions, $\mathcal{Q}$ denotes the set of rational numbers.

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

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

The output is

- set $P \subseteq U$ of parameter choices for the advisor, with $|P| \le k$,

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

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

4. AN APPROXIMATION ALGORITHM FOR LEARNING ADVISOR SETS

As Advisor Set is NP-complete, it is unlikely we can efficiently find advisor sets that are *optimal*, but we can efficiently find advisor sets that have a guarantee on how *close* they are to optimal. An *α-approximation algorithm* for a maximization problem, where $\alpha \le 1$, 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 there is a simple approximation algorithm for Advisor Set that for any constant $\ell \le k$ achieves approximation ratio at least ℓ/k when error tolerance $\epsilon = 0$.

For any 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. The procedure `Greedy`(k, ℓ) given below, which finds an approximate solution to Advisor Set, treats as global variables the universe U of parameter choices and the estimator coefficient vector c, and assumes $1 \le \ell \le k$. 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, `Greedy` outputs the best advisor set found across all cardinalities.

procedure `Greedy`(ℓ, k) **begin**
  Find an optimal subset $P \subseteq U$ of size $|P| \le \ell$
    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 $f(\widetilde{P}) \ge f(P)$ **then**
      $P := \widetilde{P}$
  end
  output P
end

We now show this natural greedy procedure is an approximation algorithm for Advisor Set. While `Greedy` finds an advisor set for *any* error tolerance $\epsilon \ge 0$, the proof of the approximation ratio we give below requires $\epsilon = 0$.

THEOREM 1. `Greedy` *is an* (ℓ/k)*-approximation algorithm for Advisor Set for constant* ℓ *and tolerance* $\epsilon = 0$.

PROOF. 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 (in which case the approximation ratio holds). So assume $\widetilde{k} \ge \ell$ (in which case $\mathcal{S}$ is nonempty). Then

$$\begin{aligned}
f(P) &\ge f(\widetilde{P}) \\
&\ge \max_{Q \in \mathcal{S}} f(Q) \qquad (5)\\
&\ge \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} f(Q) \\
&= \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} \sum_{i} w_i \, \mathcal{A}_i(Q) \\
&= \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} \sum_{i} \sum_{j \in \mathcal{O}_i(Q)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(Q)|} \\
&= \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} \sum_{j \in Q} \sum_{i \,:\, j \in \mathcal{O}_i(Q)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(Q)|} \\
&\ge \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} \sum_{j \in Q} \sum_{i \,:\, j \in \mathcal{O}_i(P^*)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(Q)|} \qquad (6)\\
&\ge \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} \sum_{j \in Q} \sum_{i \,:\, j \in \mathcal{O}_i(P^*)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(P^*)|} \qquad (7)\\
&= \frac{1}{|\mathcal{S}|} \sum_{j \in P^*} \sum_{Q \in \mathcal{S} \,:\, j \in Q} \sum_{i \,:\, j \in \mathcal{O}_i(P^*)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(P^*)|} \\
&= \frac{\binom{\widetilde{k}-1}{\ell-1}}{\binom{\widetilde{k}}{\ell}} \sum_{j \in P^*} \sum_{i \,:\, j \in \mathcal{O}_i(P^*)} \frac{w_i \, a_{ij}}{|\mathcal{O}_i(P^*)|} \\
&= \big(\ell / \widetilde{k}\big) \, f(P^*) \\
&\ge \big(\ell/k\big) \, f(P^*) \,,
\end{aligned}$$

where inequality (5) holds because $\widetilde{P}$ is an optimal set of size at most ℓ; inequality (6) holds because $Q \subseteq P^*$ implies $\big\{i \,:\, j \in \mathcal{O}_i(P^*)\big\} \subseteq \big\{i \,:\, j \in \mathcal{O}_i(Q)\big\}$ for $j \in Q$, and the terms that are lost are nonnegative; and inequality (7) holds as $\epsilon = 0$, $Q \subseteq P^*$, $j \in Q$, and $j \in \mathcal{O}_i(P^*)$ together imply $|\mathcal{O}_i(P^*)| \ge |\mathcal{O}_i(Q)|$. Thus `Greedy` achieves approximation ratio at least ℓ/k.

Finally observe that `Greedy` runs in $O(tk^2nm^\ell)$ time for t features, n benchmarks, and a universe U of m parameter choices. For constant ℓ, this is polynomial time. □

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

THEOREM 2. *Approximation ratio* ℓ/k *for* `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 $0 \le \epsilon < 1$, consider the following infinite class of instances of Advisor Set with n benchmarks, weights $w_i = 1/n$, cardinality bound $k = n$, and universe $U = \{0, 1, \ldots, n\}$ of $n{+}1$ parameter choices. The *estimator values* (which can be achieved by appropriate feature vectors F_{ij}) are: $E(A_{i0}) = 1$ for all i; $E(A_{ij}) = (1-\epsilon)/2$ for $i = j > 0$; and $E(A_{ij}) = 0$ otherwise. The *alternate alignment accuracies* are: $a_{i0} = (\ell/k) + \delta$ for all i; $a_{ij} = 1$ for $i = j > 0$; and $a_{ij} = 0$ otherwise.

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 $\widetilde{P} \supseteq \{0\}$, and hence has value $f(\widetilde{P}) = (\ell/k) + \delta$. Every greedy augmentation $P \supseteq \widetilde{P}$ also has this same value $f(P) = f(\widetilde{P})$. Thus on this instance the advisor set P output by `Greedy` has approximation ratio exactly $f(P)/f(P^*) = (\ell/k) + \delta$.

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 $(\ell/k) + \delta = \big((\ell/k) + \alpha\big)/2 < \alpha$, a contradiction. So the approximation ratio is tight. □

5. THE COMPLEXITY OF LEARNING OPTIMAL ADVISOR SETS

We now prove that Advisor Set, the problem from Section 3 of learning an optimal parameter set for an advisor, 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 $|P| \le k$ for which the objective function has value $f_c(P) \ge \ell$.

THEOREM 3. *The decision version of Advisor Set is NP-complete.*

PROOF. We use a reduction from the *Dominating Set* problem, which is NP-complete [7, 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 ($t = 1$) 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 $N(i) := \{j : (i,j) \in E\} \cup \{i\}$, 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 that 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 proves the claim.

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, and when the accuracies and feature values are all binary and benchmarks are uniformly weighted.

6. EXPERIMENTAL RESULTS

We evaluate the performance of our approach to learning parameter sets through experiments on a collection of benchmarks that are protein multiple sequence alignments. A full description of the benchmark collection, and the construction of a universe U of parameter choices that is appropriate for protein alignment, is given in [10], and is briefly summarized below. In the experiments we compare parameter advisors that use five different estimators from the literature: `MOS` [13], `PredSP` [1], `GUIDANCE` [16], `Facet` [10], and `TCS` [3].

The benchmarks suites used in our experiments consist of reference alignments that are mainly induced by performing structural alignment of the known three-dimensional structures of the proteins. Specifically we use the `BENCH` suite of Edgar [6], supplemented by a selection of benchmarks from the `PALI` suite [2]. The entire benchmark collection consists of 861 reference alignments. A subset of this collection was also used, consisting of the 605 reference alignments with at least four sequences, in order to evaluate the `GUIDANCE` estimator, which requires this many sequences in its 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*; we measured the difficulty of a benchmark by the true accuracy of the alignment of its sequences computed using the multiple alignment tool `Opal` [19, 20] under its optimal default parameter choice. We then divided the full range $[0, 1]$ of accuracies into 10 bins with difficulties $[(i-1)/10, i/10]$ for $i = 1, \ldots, 10$. The weight w_j of benchmark B_j falling in bin i that we used for training is $w_j = (1/10)(1/n_i)$, where n_i is the number of benchmarks in bin i. These weights w_j are such that each hardness bin contributes equally to the advising accuracy objective $f(P)$. Note that when the advising set P consists only of this single default parameter choice, the average advising accuracy $f(P)$ will be roughly 50%.

For each benchmark in our collection, we generated alternate alignments of its sequences using the `Opal` aligner invoked with each parameter choice from our universe U. Each *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, pair γ_E, λ_E specifies the gap-open and gap-extension penalties for *external* gaps in the alignment (also called terminal gaps), and γ_I, λ_I specifies the gap penalties for *internal* gaps (or non-terminal gaps).

The universe U of parameter choices we consider in our experiments consists of eight substitution matrices from the `BLOSUM` [8] and `VTML` [14] families combined with over 2,100 four-tuples of gap penalties that are in a range surrounding each of the default parameter values for `Opal`. This initial set of roughly 16,900 parameter choices (a substitution matrix combined with a gap-penalty assignment) was then reduced by selecting the 25 most accurate parameter choices for each of the 10 hardness bins. Unioning these top choices from all hardness bins (and removing duplicates) gave our 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 different 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 testing set and a training set of example alignments by generating $|U|$ alignments from each benchmark B in a training or test class by running `Opal` on B using each parameter choice in U. An estimator learned on the examples in the training set was evaluated on the examples in the associated test set. The results that we report are averages over twelve folds, where each fold is one of these training and testing set pairs. (Note that across these twelve folds, every example alignment is tested on exactly once.) For each fold, over 190,000 training examples were considered over the 243 parameters.

For the reduced benchmark collection used to evaluate the `GUIDANCE` estimator, we used 4-fold cross validation, performed using the same procedure described above. Each of these folds has over 109,000 training examples.

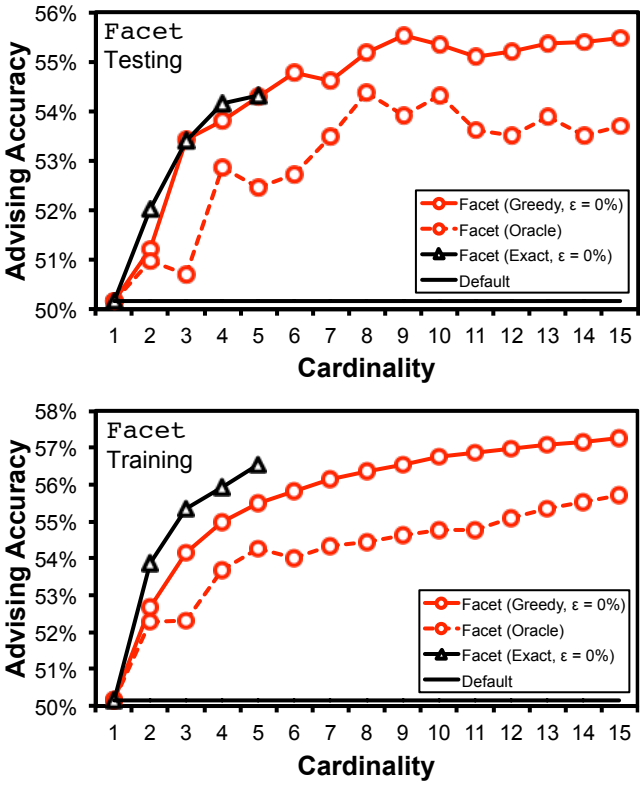


Figure 3: *Advising using exact, greedy, and oracle sets with* `Facet`. The plots show advising accuracy using the `Facet` estimator with parameter sets learned by the optimal *exact* algorithm and the *greedy* approximation algorithm for Advisor Set, and using *oracle* sets. The horizontal axis is the cardinality of the advisor set, and the vertical axis is the advising accuracy averaged across 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 top and bottom plots show accuracy on the testing and training data, respectively, where accuracies are averaged over all testing or training folds.

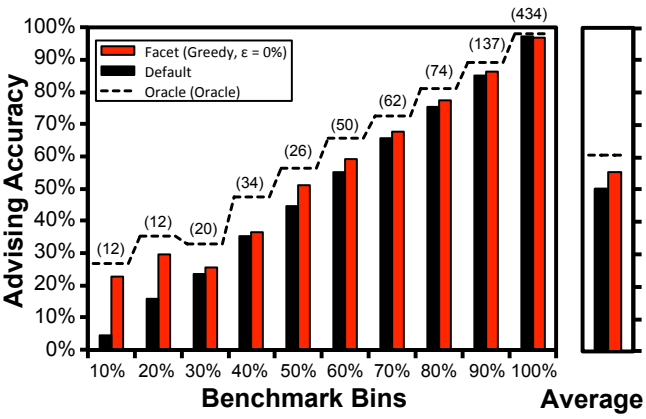


Figure 4: *Advising accuracy of* `Facet` *within benchmark bins.* In the bar chart on the left, the horizontal axis shows all ten benchmark bins, and the vertical bars show advising accuracy averaged over just the benchmarks in each bin. Black bars give the accuracy of the optimal default parameter choice, while red bars give the accuracy of advising with `Facet` using the *greedy* set of cardinality $k = 10$. The dashed line shows the limiting performance of a perfect advisor: an oracle with true accuracy as its estimator using an optimal *oracle* set of cardinality $k = 10$. The numbers in parentheses above the bars are the number of benchmarks in each bin. The bar chart on the right shows advising accuracy uniformly averaged over the bins.

6.1 Estimator features

The choice of feature functions is crucial for the accuracy of the estimator, and hence the accuracy of the resulting advisor. Estimator features should correlate with true accuracy, be efficiently computable, and be bounded in value. In practice, the strongest features use predicted secondary structure that is computed for the protein sequences; in our experiments we predicted secondary structure using `PSIPRED` [9]. Our accuracy estimator `Facet` [5, 4] uses the following five feature functions, which we briefly summarize. (Full details on the features used by `Facet` are in [10].)

- *Secondary Structure Blockiness* measures the fraction of substitutions in the alignment that are in an optimal packing of disjoint *blocks*, where a block is a subset of the alignment's rows and a consecutive interval of its columns such that all residues in the block have the same predicted secondary structure.
- *Secondary Structure Agreement* is the probability that residues paired by the substitutions in the alignment share the same secondary structure, based predicted secondary structure for the surrounding sequence.
- *Secondary Structure Identity* measures the fraction of residue pairs in substitutions that share the same predicted secondary structure.
- *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.
- *Average Substitution Score* is the average score of all substitutions in the alignment under `BLSM62` [8].

6.2 Learning advisor sets via different algorithms

We first study the accuracy of advisor sets learned by different algorithms for the `Facet` estimator. An optimal oracle set is constructed for cardinalities $1 \leq k \leq 15$ for each training instance. A coefficient vector is then found for the advisor's estimator for each of these sets by the difference-fitting method described in [10]. 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 set of size 1 (the most accurate single parameter choice) is then used as the starting point to find near-optimal *greedy* advisor sets for $k \leq 15$. Each of these advisors is then used for parameter advising in `Opal`, returning the computed alignment with the highest estimator value. The set-finding methods are compared based on the average accuracy across bins of the alignment chosen by the advisor.

Figure 3 shows the performance of these advisor sets using twelve-fold cross validation. The top plot shows advising

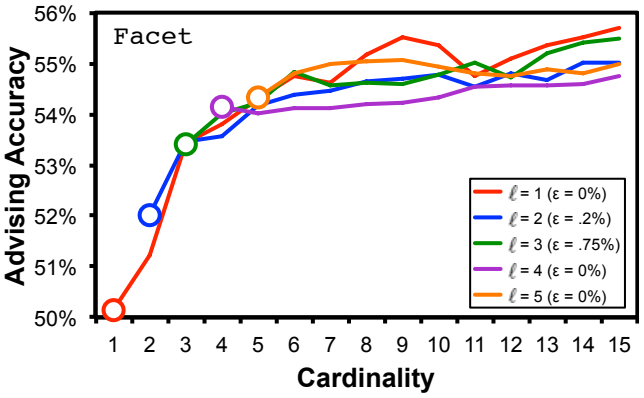


Figure 5: *Greedily augmenting exact advisor sets.* The plot shows advising accuracy using `Facet` with advisor sets learned by procedure `Greedy`, which augments an exact set of cardinality ℓ to form a larger set of cardinality $k > \ell$. Each curve is for greedily augmenting from a different exact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

accuracy on the testing data averaged over benchmarks and folds, while the bottom 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 was chosen to maximize 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 essentially performing as well as the optimal exact sets.

Figure 4 shows the performance within each benchmark bin when advising with `Facet` using greedy sets of cardinality $k = 10$. 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 the approximation algorithm `Greedy`, we tested the advising accuracy of the resulting greedy sets learned using cardinalities $1 \leq \ell \leq 5$. Figure 5 shows the accuracy of the resulting advisor using the greedy sets of cardinalities $\ell < k \leq 15$ augmented from exact sets of cardinalities $1 \leq \ell \leq 5$. The points with circles show the accuracy of the optimal exact set that is used within procedure `Greedy` for augmentation.

Notice that this initial optimal set size ℓ has little effect on the accuracy of the resulting advisor; at most cardinalities, starting from the single best parameter choice (corresponding to $\ell = 1$) has highest advising accuracy. This is likely due to the behavior noted earlier in Figure 3 that exact sets do not generalize as well as greedy sets.

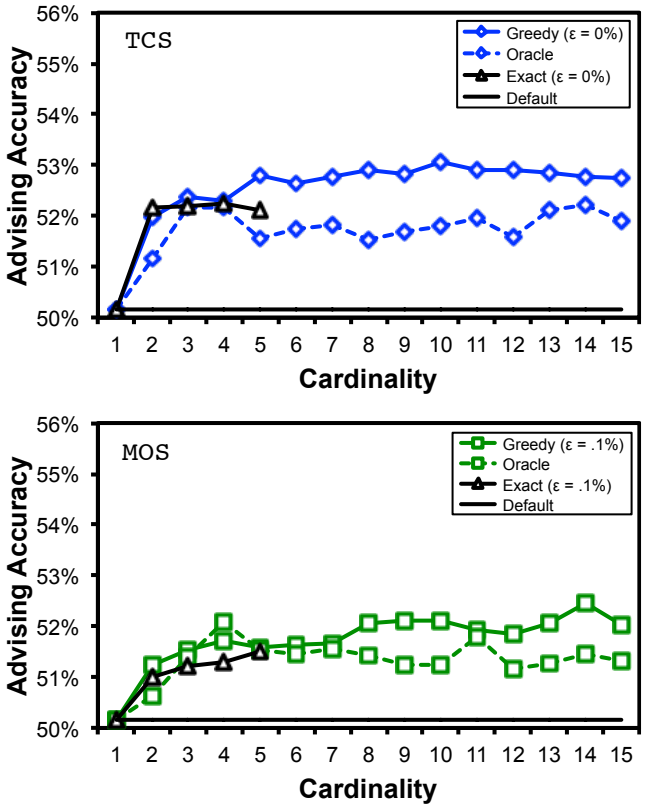


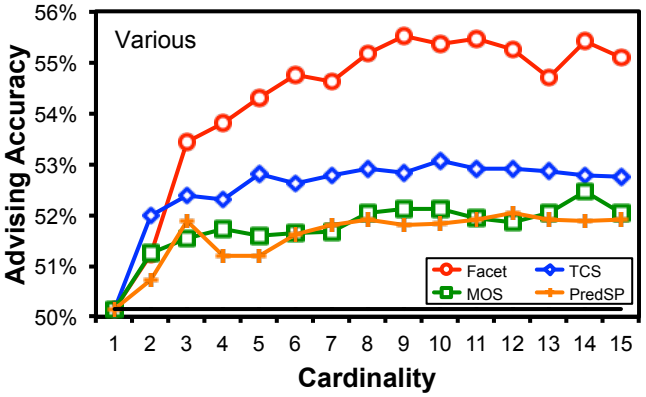
Figure 6: *Advising using exact, greedy and oracle sets with* `TCS` *and* `MOS`. The plots show advising accuracy on testing data using the `TCS` and `MOS` estimators with parameter sets learned for these estimators by the exact and greedy algorithms for Advisor Set, and using oracle sets, as in Figure 3. The top and bottom plots are for the `TCS` and `MOS` estimators, respectively.

6.4 Learning advisor sets for other estimators

We also learned advisor sets for other accuracy estimators besides `Facet`: namely `TCS`, `MOS`, `PredSP`, and `GUIDANCE`. The scoring-function-based accuracy estimators `TCS`, `PredSP`, and `GUIDANCE` do have any dependence on the advisor set cardinality or the training benchmark sets used. The support-based estimator `MOS`, however, requires a set of alternate alignments in order to compute its estimator value or an alignment. For each experiment, an alignment's `MOS` value was computed using alternate alignments generated by aligning under the parameter choiceds in the oracle set; if the parameter choice being tested is 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 $k \leq 15$ (which augmented 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 6 shows the performance on testing data of the greedy, exact, and oracle advisor sets learned for the best two other estimators, `TCS` and `MOS`. The results of finding a greedy advisor set for `TCS` for cardinalities larger than 5 are similar to those seen for `Facet` (there is a



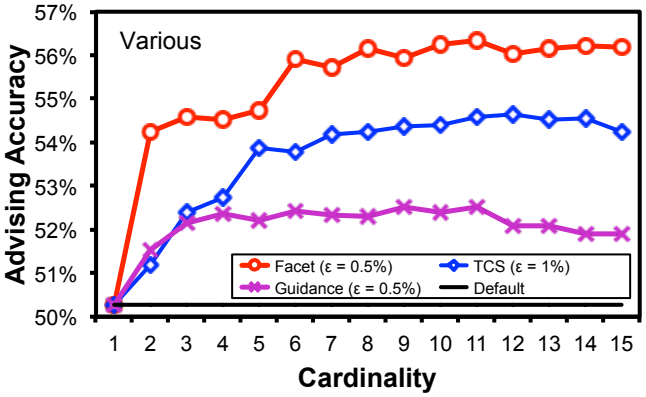
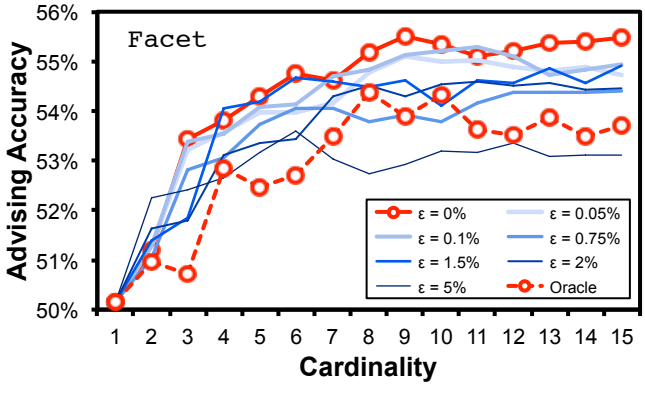


Figure 7: *Comparing estimators on greedy advisor sets.* The plots show advising accuracy on greedy sets learned for the following estimators from the literature: `Facet` [10], `TCS` [3], `MOS` [13], `PredSP` [1], and `GUIDANCE` [16]. The vertical axis is advising accuracy on the testing data, averaged 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 top plot uses the full suite of 861 benchmarks. The bottom plot includes the `GUIDANCE` estimator, which requires alignments with at least four sequences; this plot uses a reduced suite of the 605 benchmarks satisfying this requirement.



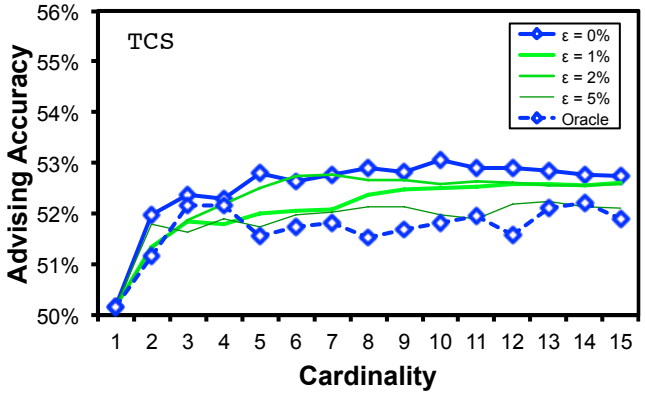


Figure 8: *Effect of error tolerance on advising accuracy.* The plots show advising accuracy on testing data using greedy sets learned for the two best estimators, at various error tolerances $\epsilon \geq 0$. The top and bottom plots are for the `Facet` and `TCS` estimators, respectively. For comparison, both plots include a curve showing performance using the estimator on oracle sets, drawn with a dashed line. The solid curves with circles or diamonds highlight the best overall error tolerance.

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 then the oracle set.

In addition to `TCS` and `MOS`, performance of the greedy advisor sets learned for the `PredSP` and `GUIDANCE` estimators are shown in Figure 7. The bottom plot shows advising accuracy for `Facet`, `TCS`, and `GUIDANCE` on the subset of benchmarks that have at least four sequences. Notice that while advising with each of these estimators tends to eventually reach a plateau of advising accuracy, the advising performance does always improve when using advising sets with more than simply a single default parameter choice. Notice also that the plateau for `Facet` (the topmost curve in the plots) occurs at the greatest cardinality and accuracy.

6.5 Varying the error tolerance

In all preceding experiments, an error tolerance ϵ was always used that gave the most accurate advisor on the testing data. Prior to conducting these experiments, our intuition was that increasing the tolerance ϵ should improve the generalization of an advisor set. Figure 8 shows the effect of different ϵ values on the testing accuracy of an advisor. No clear relationship between testing accuracy and tolerance ϵ is evident, though for the `Facet` and `TCS` estimators, setting $\epsilon = 0$ does generally give the best overall advising accuracy.

6.6 Parameter sets

Table 1 gives sets of parameter choices for the `Opal` aligner for cardinalities $k \leq 10$, found by the *greedy* approximation algorithm for the `Facet` estimator, for one fold of training data. In the table, The greedy set of cardinality i is the parameter choices at rows 1 through i. Here a parameter choice is 5-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 the range $[0, 100]$. The accuracy column gives the advising accuracy using the greedy set (in `Opal` with `Facet`) on the *training* data, uniformly averaged over the benchmark bins. This averaging will tend to yield accuracies close to 50%.

Table 1: *Greedy Parameter Sets for* `Opal` *Using* `Facet`

Cardinality	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%

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

This work was supported by US National Science Foundation Grant IIS-1217886 to J.K., and a PhD fellowship to D.D. through US National Science Foundation Grant DGE-0654435.

9. REFERENCES

[1] V. Ahola, T. Aittokallio, M. Vihinen, and E. Uusipaikka. Model-based prediction of sequence alignment quality. *Bioinformatics*, 24(19):2165–2171, Sept. 2008.

[2] 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*, 29(1):61–65, Jan. 2001.

[3] 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*, 31(6):1625–1637, Apr. 2014.

[4] 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.

[5] 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)*, pages 45–59, 2012.

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

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

[8] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences USA*, 89(22):10915–10919, Nov. 1992.

[9] D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292(2):195–202, Sept. 1999.

[10] J. Kececioglu and D. DeBlasio. Accuracy estimation and parameter advising for protein multiple sequence alignment. *Journal of Computational Biology*, 20(4):259–279, Apr. 2013.

[11] J. Kim and J. Ma. `PSAR`: measuring multiple sequence alignment reliability by probabilistic sampling. *Nucleic Acids Research*, 39(15):6359–6368, Aug. 2011.

[12] G. Landan and D. Graur. Heads or tails: a simple reliability check for multiple sequence alignments. *Molecular Biology and Evolution*, 24(6):1380–1383, 2007.

[13] T. Lassmann and E. L. L. Sonnhammer. Automatic assessment of alignment quality. *Nucleic Acids Research*, 33(22):7120–7128, Dec. 2005.

[14] 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*, 19(1):8–13, Jan. 2002.

[15] C. Notredame, L. Holm, and D. G. Higgins. `COFFEE`: an objective function for multiple sequence alignments. *Bioinformatics*, 14(5):407–422, June 1998.

[16] 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*, 27(8):1759–1767, July 2010.

[17] 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*, 314(4):937–951, Dec. 2001.

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

[19] 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*, 23(13):i559–i568, July 2007.

[20] T. J. Wheeler and J. D. Kececioglu. `Opal`: software for aligning multiple biological sequences, Version 2.1.0. `http://opal.cs.arizona.edu`, 2012.