Learning Scoring Schemes for Sequence Alignment from Partial Examples

Eagu Kim and John Kececioglu

Abstract—When aligning biological sequences, the choice of scoring scheme is critical. Even small changes in gap penalties, for example, can yield radically different alignments. A rigorous way to learn parameter values that are appropriate for biological sequences is through *inverse parametric sequence alignment*. Given a collection of examples of biologically correct reference alignments, this is the problem of finding parameter values that make the scores of the reference alignments be as close as possible to those of optimal alignments of their sequences. We extend prior work on inverse parametric alignment to *partial examples*, which contain regions where the reference alignment is not specified, and to an improved formulation based on minimizing the *average error* between the scores of the reference alignments and the scores of optimal alignments. Experiments on benchmark biological alignments show we can learn scoring schemes that generalize across protein families, and that boost the accuracy of multiple sequence alignment by as much as 25 percent.

Index Terms—Inverse parametric sequence alignment, substitution score matrices, affine gap penalties, linear programming, cutting plane algorithms.

1 Introduction

A fundamental question that arises whenever biological sequences are aligned is to decide what parameter values to use for the alignment scoring scheme. For example, the standard scoring function for protein sequence alignment requires determining values for 210 substitution scores and two gap penalties. In practice, substitution scores are usually set by convention to a matrix from the PAM [5] or BLOSUM [17] families, while gap penalties are often set by hand through trial and error. Despite this practice, choosing appropriate values for these parameters is critical. Even small changes in gap penalties can yield radically different alignments, while aligning a specific family of sequences may require substitution scores that differ substantially from a conventional matrix.

A rigorous approach to determining these values is through *inverse parametric sequence alignment* [16], [31], which sets parameters using examples of biologically correct reference alignments. Informally, inverse alignment seeks parameter values for the alignment scoring function that make the examples be optimal alignments of their sequences. Put another way, it seeks a parameter choice at which the examples are optimal solutions to the sequence alignment problem. Note that if a parameter choice exists that makes every example be the unique-optimal alignment of its sequences, then any algorithm that computes optimal sequence alignments will perfectly recover the examples, given the example sequences and this parameter choice as inputs. In general, inverse alignment belongs to the broad area of inverse parametric optimization [9], where examples of optimal solutions to a combinatorial optimization problem are used to set parameter values in its objective function.

In practice, a biologically useful parameter choice rarely exists that makes a collection of example reference alignments all be optimal [19] (let alone unique-optimal). Consequently, the problem becomes one of finding parameter values that make the examples score as close as possible to optimal. An important issue in formulating the problem is deciding what measure of *error* to minimize between the scores of the examples and the scores of optimal alignments, when assessing closeness to optimality.

Recently, Kececioglu and Kim [19] discovered a new method for inverse alignment based on linear programming that for the first time could quickly learn best possible values for all 212 parameters in the standard model for global protein sequence alignment from hundreds of examples of complete reference alignments. Their approach minimized the maximum relative error in scores across the examples. In this paper, we extend this work in several directions:

- to *partial examples*, which contain regions where the reference alignment is not specified,
- to an improved error model involving minimization of the *average absolute error* across the examples,
- to a stronger approach for eliminating *degenerate solutions* that are not biologically useful, and
- to an experimental study of the *recovery rate* of the learned scoring schemes when used for *multiple sequence alignment*.

The issue of partial versus complete examples is especially important when the reference alignments are drawn from suites of benchmark protein multiple alignments that have been determined by aligning the three-dimensional structures of the protein molecules [24], [1], [2], [33]. While such benchmarks provide the most accurate protein alignments currently available, the regions where these reference alignments are reliable usually consist of a series of aligned conserved blocks that rarely cover the entire sequences;

• *The authors are with the Department of Computer Science, The University of Arizona, Tucson, AZ 85721. E-mail: {egkim, kece}@cs.arizona.edu.*

Manuscript received 7 Dec. 2007; revised 26 Apr. 2008; accepted 21 May 2008; published online 3 June 2008.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBBSI-2007-12-0165.
Digital Object Identifier no. 10.1109/TCBB.2008.57.

between these blocks, the alignment is often unreliable or unspecified.

The improved approach to inverse alignment that results from these extensions, which minimizes average absolute error given partial examples, remains fast and achieves better recovery. To briefly summarize the results, the recovery of reference alignments by the learned scoring schemes improves upon conventional amino acid substitution matrices by as much as 25 percent when used for multiple alignment of specific sequence families. Learning a scoring scheme from more than 200 partial examples, whose reference alignments cover about 40 percent of each sequence on average, takes around 4 hours on a laptop.

1.1 Related Work

Inverse parametric sequence alignment was introduced in the seminal paper of Gusfield and Stelling [16]. They considered the problem for the case of two parameters and one example, and sought parameter values that made the example be an optimal scoring alignment of its sequences. For this form of inverse alignment, they presented an indirect approach that attempted to avoid computing the complete parametric decomposition [34], [15], [27] of the parameter space. Sun et al. [31] gave the first direct algorithm for inverse alignment for the case of three parameters and one example. Their algorithm, which is quite involved, finds values for the three parameters that make the example alignment score optimal in $O(n^2 \log n)$ time, for two strings of length n.

In a significant advance, Kececioglu and Kim [19], using a completely different approach based on linear programming, gave the first polynomial-time algorithm for arbitrarily many parameters and examples. Their approach is also flexible, and solves the unique-optimal and near-optimal variations of inverse alignment as well. For the near-optimal variation, their algorithm finds a parameter choice that makes the examples score as close as possible to optimal in terms of minimizing the maximum relative error of the examples. As they demonstrated, it is also fast in practice.

The approach to inverse alignment developed in [19] actually solves the more general problem of *inverse parametric optimization*. For a combinatorial optimization problem $\mathcal{P}$, inverse parametric optimization seeks parameter values for the objective function of $\mathcal{P}$ that make a given example solution to an instance $\mathcal{I}$ of $\mathcal{P}$ be an optimal solution to instance $\mathcal{I}$. Their approach solves the inverse parametric problem in polynomial time for any problem $\mathcal{P}$ where: 1) the objective function for $\mathcal{P}$ is linear in its parameters, and 2) problem $\mathcal{P}$ can be solved in polynomial time for any fixed choice of its parameters. This solves inverse parametric optimization for an extremely wide range of problems $\mathcal{P}$, including many classic combinatorial problems such as sequence alignment, shortest paths, spanning trees, network flow, and maximum matchings. The authors recently learned that Eppstein [9] discovered a similar approach to general inverse parametric optimization. Eppstein applied it in the context of minimum spanning trees to find edge weights that make an example tree be the unique optimal spanning tree. In the context of biological sequence alignment, this unique-optimal form of the inverse problem rarely has a solution [19].

Alternative approaches for determining alignment parameters have been recently proposed based on machine learning. Do et al. [7] use discriminative training on conditional random fields to find parameter values for a hidden Markov model of sequence alignment. Their approach requires solving a convex nonlinear numerical optimization problem that becomes nonconvex in the presence of examples that are partial alignments, and is not guaranteed to run in polynomial time. Yu et al. [37] describe a support vector machine approach for learning parameters to align a protein sequence to a protein structure. Their approach involves solving a quadratic numerical optimization problem with linear constraints, and for the first time incorporates a measure of alignment recovery directly into the problem formulation. Both of these approaches require considerable computational resources, appearing to take days on large instances to compute an approximate solution.

In contrast to these machine learning approaches, our method for inverse alignment only involves linear programming, for which an optimal solution can be computed quickly, even for very large instances. We also rigorously address for the first time the issue of input examples that are partial alignments.

1.2 Overview

The next section presents several variations of inverse alignment, with relative or absolute error, and with complete or partial examples. Section 3 reduces these variations to linear programming, and develops an iterative approach to partial examples. Finally, Section 4 presents results from experiments on recovering benchmark protein alignments when using learned parameters for both pairwise and multiple sequence alignment.

2 Inverse Alignment and Its Variations

The conventional sequence alignment problem is: given a pair of sequences and a scoring function f on alignments, find an alignment $\mathcal{A}$ of the sequences that has an optimal score under f. The *inverse alignment problem* turns this around: given an example alignment $\mathcal{A}$ of a pair of sequences, find parameter values for scoring function f that make $\mathcal{A}$ be an optimal alignment of its sequences. To learn parameter values that are useful in practice for biologists, this basic form of inverse alignment, which was first studied in [16] and [31], must be generalized considerably: to multiple examples, to suboptimal alignments, to partial examples, and to handle degeneracy, each of which we consider in turn.

When function f has many parameters, many example alignments $\mathcal{A}$ are needed to determine reliable values for the parameters. Accordingly, we take the input to inverse alignment to be a large *collection* of example alignments. In practice, there are usually no biologically useful parameter values that make all the example alignments have optimal score. Consequently, we consider finding parameter values that make the examples score as close as possible to optimal, and we examine two optimization criteria for measuring the *error* between the example scores and the optimal alignment scores: minimizing relative error or absolute error. Finally, the type of benchmark reference alignments that are available in practice for learning parameters often consist of regions where the alignment is specified, interspersed with regions where no alignment is specified. We call such a

reference alignment a *partial example*, since it is only a partial alignment of its sequences. When the reference specifies a complete alignment of its sequences, we call it a *complete example*.

Our approach to inverse alignment from partial examples builds upon a solution to the problem with complete examples, which we discuss first.

2.1 Complete Examples

Inverse alignment from complete examples with arbitrarily many parameters was first considered by Kececioglu and Kim [19]. They examined the relative error criterion, which we review below.

Let f be the alignment *scoring function*, which gives score $f(\mathcal{A})$ to alignment $\mathcal{A}$. Typically, f is a function of several *parameters* $p_1, p_2, \ldots, p_t$, which assign scores or penalties to various alignment features such as substitutions and gaps. (For example, the standard scoring model for aligning protein sequences has 210 substitution scores for all unordered pairs of amino acids, plus two gap penalties for opening and extending a gap, for a total of $t = 212$ parameters.) We view the entire set of parameters as a vector $p = (p_1, \ldots, p_t)$. When we want to emphasize the dependence of f on its parameters p, we write f_p.

The input consists of many example alignments $\mathcal{A}_i$, where each example aligns a corresponding set of sequences $\mathcal{S}_i$. (Typically, the examples $\mathcal{A}_i$ are induced pairwise alignments that come from a structural multiple alignment; in this case, each $\mathcal{S}_i$ contains two sequences.) For scoring function f and parameters p, we write $f_p^*(\mathcal{S}_i)$ for the score of an *optimal* alignment of sequences $\mathcal{S}_i$ under f_p. We assume that an optimal alignment *minimizes* f.

The following defines inverse alignment under the relative error criterion assuming the input contains complete examples.

Definition 1 (Inverse Alignment under Relative Error). *Inverse Alignment from complete examples under the relative error criterion is the following problem. Let the alignment scoring function be f_p with parameter vector $p = (p_1, \ldots, p_t)$ drawn from domain $\mathcal{D}$. The* input *is a collection of complete alignments $\mathcal{A}_1, \ldots, \mathcal{A}_k$ that respectively align the sets of sequences $\mathcal{S}_1, \ldots, \mathcal{S}_k$. The* output *is parameter vector*

$$x^* \;:=\; \operatorname*{arg\,min}_{x \in \mathcal{D}} \; E_{\mathrm{rel}}(x),$$

where

$$E_{\mathrm{rel}}(x) \;:=\; \max_{1 \le i \le k} \frac{f_x(\mathcal{A}_i) - f_x^*(\mathcal{S}_i)}{f_x^*(\mathcal{S}_i)}.$$

In other words, the output vector x^* minimizes the maximum relative error of the alignment scores of the examples.

An underlying issue in this formulation of inverse alignment is that the problem is *degenerate*. The *trivial* parameter choice $x = (0, \cdots, 0)$ makes every alignment score the same, thus making each example be an optimal alignment. This undesirable solution must be ruled out, though how this is done depends on the particular form of alignment being considered. Section 3.3 presents a new approach for eliminating a general family of degenerate solutions that applies to global sequence alignment [14]. This general family includes the trivial solution as a special case.

When scoring function f_p is linear in its parameters p, inverse alignment under relative error can be solved in polynomial time [19] as long as an optimal alignment can be computed in polynomial time for any *fixed* parameter choice. We review the solution in Section 3, which uses a reduction to linear programming. The above formulation considers the *maximum error* of the examples, because as we will see later in Section 3.1.1, minimizing the average relative error would lead to an optimization problem with nonlinear constraints.

We also consider here a new model: minimizing the absolute error. This has the advantage that we can minimize the *average error* of the examples and still have a formulation that is efficiently solvable by linear programming.

The following defines inverse alignment under the absolute error criterion again assuming complete examples.

Definition 2 (Inverse Alignment under Absolute Error). *Inverse Alignment from complete examples under the absolute error criterion is the following problem. The input is a collection of complete alignments $\mathcal{A}_i$ of sequences $\mathcal{S}_i$ for $1 \le i \le k$. The output is parameter vector*

$$x^* \;:=\; \operatorname*{arg\,min}_{x \in \mathcal{D}} \; E_{\mathrm{abs}}(x),$$

where

$$E_{\mathrm{abs}}(x) \;:=\; \frac{1}{k} \sum_{1 \le i \le k} \Big(f_x(\mathcal{A}_i) \;-\; f_x^*(\mathcal{S}_i) \Big).$$

Output vector x^* minimizes the average absolute error of the alignment scores of the examples.

Note that this formulation is also degenerate, which is addressed in Section 3.3.

We next extend to input examples that are partial alignments.

2.2 Partial Examples

For inverse alignment of protein sequences, the best example alignments that are available come from multiple alignments of protein families that are determined by aligning the three-dimensional structures of family members. Several suites of such benchmark alignments are currently available [24], [1], [2], [33] and are widely used for evaluating the accuracy of software for multiple alignment of protein sequences. Most all these benchmarks, however, are partial alignments. The benchmark alignment has regions that are reliable and where the alignment is specified, but between these regions, the alignment is effectively left unspecified. These reliable regions are usually the *core blocks* of the multiple alignment, which are sections of the alignment where structure is conserved across the family. Core blocks are typically gap free, though they can sometimes contain gaps.

For our purposes, a *partial example* is an alignment $\mathcal{A}$ of sequences $\mathcal{S}$ where each column of $\mathcal{A}$ is labeled as being either reliable or unreliable. A *complete example* is a partial example whose columns are all labeled reliable.

Note that in this definition, a partial example may label a gap column as reliable. Consequently, partial examples do not simply specify which pairs of positions to align by substitutions, but may also specify which positions should be in gaps.

When learning parameters by inverse alignment from partial examples, we treat the unreliable columns as missing information: such columns do not specify the alignment of the example sequences. Given a partial example $\mathcal{A}$ for sequences $\mathcal{S}$, a *completion* $\overline{\mathcal{A}}$ is a complete example for $\mathcal{S}$ that agrees with the reliable columns of $\mathcal{A}$. In other words, a completion $\overline{\mathcal{A}}$ can change $\mathcal{A}$ on the substrings that are in unreliable columns, but must not alter $\mathcal{A}$ in reliable columns. Note that in general, a partial example $\mathcal{A}$ does not have a unique completion $\overline{\mathcal{A}}$.

We define inverse alignment from partial examples as the problem of finding the optimal parameter choice over all possible completions of the examples.

Definition 3 (Inverse Alignment from Partial Examples). *Inverse Alignment from partial examples is the following problem. The input is a collection of partial alignments $\mathcal{A}_i$ of sequences $\mathcal{S}_i$ for $1 \leq i \leq k$. The output is parameter vector*

$$x^* \;:=\; \arg\min_{x \in \mathcal{D}} \; \min_{\overline{\mathcal{A}}_1, \ldots, \overline{\mathcal{A}}_k} \; E(x),$$

where error function E is either E_{abs} or E_{rel}, and symbol $\overline{\mathcal{A}}_i$ varies over all possible completions of $\mathcal{A}_i$. In other words, vector x^* minimizes the error of the example scores over all possible completions of the partial examples.

We make a few remarks on this definition. First, this formulation finds parameter values for which optimally aligning the unreliable regions yields a completion that scores as close as possible to an optimal unconstrained alignment of the example sequences. Second, even when the reliable regions are all gap free (which is common when partial examples are obtained from the standard suites of benchmark protein alignments), this formulation still learns useful values for gap penalties, as appropriate values for gaps in the unreliable regions are necessary for the reliable regions to be part of alignments that have a close-to-optimal score. Third, note that this formulation for partial examples contains the prior formulations on complete examples as special cases.

In the next section, we reduce inverse alignment from complete examples to linear programming, and tackle partial examples by solving a series of problems on complete examples.

3 Solution by Linear Programming

When the alignment scoring function f_p is linear in its parameters p, inverse alignment from complete examples under relative error can be reduced to linear programming [19], and a similar reduction applies to absolute error. We define a linear scoring function as follows: Suppose f scores an alignment $\mathcal{A}$ by measuring $t+1$ features of $\mathcal{A}$ through functions $g_0, g_1, \ldots, g_t$ and combines these measures into one score through a weighted sum involving parameter vector $p = (p_1, \ldots, p_t)$ by

$$f_p(\mathcal{A}) \;:=\; g_0(\mathcal{A}) \;+\; \sum_{1 \leq i \leq t} p_i \; g_i(\mathcal{A}).$$

Then, we say f is *linear* in parameters $p_1, \ldots, p_t$. (The initial term given by function g_0 usually arises when some parameters in the scoring function are fixed, such as when learning gap penalties for a known substitution scoring matrix [16], [31], [19], in which case $g_0(\mathcal{A})$ measures the total substitution score of alignment $\mathcal{A}$. When all parameters are varying, the term for g_0 is typically zero.)

For example, in the standard scoring model for global alignment of protein sequences, there is a substitution score σ_{ab} for every unordered pair a, b of amino acids, a penalty γ for opening a gap, and a penalty λ for extending a gap. (A *gap* in an alignment is a maximal run of either insertion or deletion columns. A gap of ℓ columns has cost $\gamma + \lambda\ell$, which is often called an *affine* gap cost.) This gives a scoring function with 212 parameters σ_{ab}, γ, and λ for the alphabet of 20 amino acids. For this scoring model, the functions $g_{\sigma,a,b}$ count the number of substitutions of each type a, b in $\mathcal{A}$, and functions g_γ and g_λ respectively count the number of gaps and the total length of all gaps in $\mathcal{A}$.

We first describe the linear programming approach for complete examples, and then discuss its extension to partial examples.

3.1 Complete Examples

As described in detail in [19], inverse alignment from complete examples with relative error can be reduced to solving a series of linear programs. We briefly review this solution for relative error, and then show how to modify it for absolute error.

For the standard model of protein sequence alignment, the linear programs have the following form. There is a variable for each parameter σ_{ab}, γ, and λ in the alignment scoring function f. The domain $\mathcal{D}$ of these parameters is described by the following *domain inequalities*.

When the alignment problem is to minimize the score f of a global alignment, parameter values can always be shifted so they are nonnegative and then scaled so the largest magnitude is 1, without changing the problem. Thus, domain $\mathcal{D}$ has inequalities

$$(0,0,0) \;\leq\; (\sigma_{ab}, \gamma, \lambda) \;\leq\; (1,1,1). \tag{1}$$

The description of $\mathcal{D}$ also has the inequalities

$$\sigma_{aa} \;\leq\; \sigma_{ab}, \tag{2}$$

since an identity should cost no more than a substitution involving that letter. (As an aside, inequality (2) holds for PAM [5] and BLOSUM [17] similarity scores when they are transformed into substitution costs.)

The remaining inequalities, which ensure that the examples score as close as possible to optimal alignments, differ for the relative and absolute error criteria.

3.1.1 Relative Error

For the relative error criterion, we first consider the problem assuming a fixed upper bound ϵ on the relative error. For a given bound ϵ, we test whether there is a feasible solution x with relative error at most ϵ by solving a linear program. We then find the smallest ϵ^*, to within a specified precision, for which there is a feasible solution, using binary search on ϵ. With precision ξ, this binary search for ϵ^* involves solving $O(\log(\epsilon^*/\xi))$ linear programs. The feasible solution x^* found at bound ϵ^* is then an optimal solution to inverse alignment under relative error.

Beyond the domain inequalities, the remaining inequalities in each linear program enforce that the relative error of all examples is at most ϵ. For each example $\mathcal{A}_i$, and *every*

alignment $\mathcal{B}_i$ of sequences $\mathcal{S}_i$, the linear program for ϵ has an inequality

$$f_x(\mathcal{A}_i) \;\leq\; (1+\epsilon)\; f_x(\mathcal{B}_i). \tag{3}$$

Notice that for a fixed value of ϵ, this is a linear inequality in parameters x, since function f_x is linear in x. Example $\mathcal{A}_i$ satisfies all these inequalities iff the inequality with $\mathcal{B}_i = \mathcal{B}^*$ is satisfied, where $\mathcal{B}^*$ is an optimal-scoring alignment of $\mathcal{S}_i$ under parameters x. In other words, the inequalities are all satisfied iff the score of $\mathcal{A}_i$ has relative error at most ϵ under f_x. Finding the minimum ϵ for which this system of inequalities has a feasible solution x corresponds to minimizing the *maximum* relative error of the example scores.

As an aside, one could attempt to minimize the *average* relative error across the examples by modifying this approach as follows: Each example $\mathcal{A}_i$ would have a corresponding error variable ϵ_i, and inequality (3) would be modified by replacing the global constant ϵ with the variable ϵ_i. The objective function would be to minimize $\frac{1}{k}\sum_{1\leq i\leq k}\epsilon_i$. Unfortunately, inequality (3) then becomes quadratic in variable ϵ_i and the vector of variables x, which is no longer solvable by linear programming.

The above linear program given by inequalities (1), (2), and (3) has an exponential number of inequalities of type (3), since for an example $\mathcal{A}_i$, the number of alignments $\mathcal{B}_i$ of $\mathcal{S}_i$ is exponential in the lengths of the sequences [11]. Nevertheless, this program can be solved in *polynomial time* [19] using a far-reaching result from linear programming theory known as the equivalence of optimization and separation [13]. This equivalence result states that one can solve a linear program in polynomial time iff one can solve the *separation problem* for the linear program in polynomial time [12], [28], [18]. The separation problem is, given a possibly infeasible vector $\widetilde{x}$ of parameter values, to report an inequality from the linear program that is violated by $\widetilde{x}$, or to report that $\widetilde{x}$ satisfies the linear program if there is no violated inequality.

We can solve the separation problem in polynomial time for the above linear program by the following algorithm. Given a vector $\widetilde{x}$ of parameter values, for each example $\mathcal{A}_i$ we compute an optimal-scoring alignment $\mathcal{B}^*$ of $\mathcal{S}_i$ under f using $\widetilde{x}$. If inequality (3) is satisfied when $\mathcal{B}_i = \mathcal{B}^*$, then the inequalities are satisfied for all $\mathcal{B}_i$; on the other hand, if inequality (3) is not satisfied for $\mathcal{B}^*$, this gives the requisite violated inequality. For a problem with k examples, solving the separation problem involves computing at most one optimal alignment for each example, for a total of at most k optimal alignments, which runs in polynomial time.

As a consequence, the equivalence of separation and optimization implies that the full linear program can be solved in polynomial time. This should mainly be viewed, however, as an existence proof for a polynomial-time algorithm for inverse alignment. The equivalence theorem relies on the ellipsoid method for linear programming (which is slow in practice), and does not provide a good bound on the polynomial for the running time of the resulting algorithm [13].

In practice, a polynomial-time solution to the separation problem is typically leveraged in the following *cutting plane algorithm* [4] for solving a linear program with inequalities $\mathcal{L}$.

1. Start with a small subset $\mathcal{P}$ of the inequalities in $\mathcal{L}$.
2. Compute an optimal solution $\widetilde{x}$ to the linear program given by subset $\mathcal{P}$. If no such solution exists, halt and report that $\mathcal{L}$ is infeasible.
3. Call the separation algorithm for $\mathcal{L}$ on $\widetilde{x}$. If the algorithm reports that $\widetilde{x}$ satisfies $\mathcal{L}$, output $\widetilde{x}$ and halt: $\widetilde{x}$ is an optimal solution for $\mathcal{L}$.
4. Otherwise, add to $\mathcal{P}$ the violated inequality returned by the separation algorithm in step 3, and loop back to step 2.

While such cutting plane algorithms are not guaranteed to terminate in polynomial time, they can be fast in practice [19].

For inverse alignment, we start with subset $\mathcal{P}$ initialized to the domain inequalities (1) and (2), together with the nondegeneracy inequality (5) described later in Section 3.3.

3.1.2 Absolute Error

For the absolute error criterion, we modify the linear program as follows: For each example $\mathcal{A}_i$, we have an additional error variable δ_i. Inequality (3) for each example $\mathcal{A}_i$ is replaced by

$$f_x(\mathcal{A}_i) \;\leq\; f_x(\mathcal{B}_i) \;+\; \delta_i. \tag{4}$$

The objective function for the linear program is to minimize

$$\frac{1}{k}\sum_{1\leq i\leq k}\delta_i.$$

Notice that minimizing this objective function forces each δ_i to have the value

$$\delta_i \;=\; f_x(\mathcal{A}_i) \;-\; f_x^*(\mathcal{S}_i).$$

Thus, an optimal solution x^* to this linear program with inequalities (1), (2), and (4) gives a parameter vector that minimizes the average absolute error of the example scores. Again the program has exponentially many inequalities of type (4), but the same separation algorithm that computes an optimal alignment $\mathcal{B}^*$ solves the separation problem in polynomial time, so in principle the linear program can be solved in polynomial time. In practice, we use a cutting plane algorithm as described above.

3.2 Partial Examples

Inverse alignment from partial examples involves optimizing over all possible completions of the examples. While for partial examples we do not know how to efficiently find an optimal solution, we present a practical iterative approach which as demonstrated in Section 4 finds a good solution.

We start with an initial completion $\overline{\mathcal{A}}_i^{(0)}$ for each partial example $\mathcal{A}_i$. This initial completion may be formed by optimally aligning each unreliable region of $\mathcal{A}_i$ using a default parameter choice $x^{(0)}$; we call this the *default* initial completion. (In practice, for $x^{(0)}$ we use a standard substitution matrix [17] with appropriate gap penalties.) Alternatively, an initial completion may be obtained by simply using the complete alignment given by both the unreliable and reliable columns of the partial example; we call this the *trivial* initial completion.

We then iterate the following process for $j = 0, 1, 2, \ldots$. Compute an optimal parameter choice $x^{(j+1)}$ by solving the inverse alignment problem on the complete examples

$\overline{\mathcal{A}}_i^{(j)}$. Given $x^{(j+1)}$, form a new completion $\overline{\mathcal{A}}_i^{(j+1)}$ of each example $\mathcal{A}_i$ by:

1. computing an alignment of each unreliable region that is optimal with respect to parameters $x^{(j+1)}$, and
2. concatenating these alignments of the unreliable regions, alternating with the alignments given by the reliable regions, to form a new complete example $\overline{\mathcal{A}}_i^{(j+1)}$.

Such a completion optimally stitches together the reliable regions of the partial example, using the current estimate for parameter values.

This *iterative scheme* alternates a step of finding optimal parameters with a step of finding optimal completions, yielding a sequence of parameters and completions:

$$\overline{\mathcal{A}}^{(0)} \;\mapsto\; x^{(1)} \;\mapsto\; \overline{\mathcal{A}}^{(1)} \;\mapsto\; x^{(2)} \;\mapsto\; \cdots.$$

As the following shows, the error of successive parameter estimates decreases monotonically.

Theorem 1 (Error Convergence for Partial Examples). *For the iterative scheme for inverse alignment from partial examples, denote the error in score for iteration $j \geq 1$ by*

$$e_j \;:=\; E\Big(x^{(j)},\, \overline{\mathcal{A}}^{(j-1)}\Big),$$

where the right-hand side measures error criterion E_{abs} or E_{rel} for the given parameters on the given completions. Then,

$$e_1 \;\geq\; e_2 \;\geq\; e_3 \;\geq\; \cdots \;\geq\; e^*,$$

where e^* is the optimum error for inverse alignment from partial examples $\mathcal{A}_i$ under criterion E.

Proof. Since $\overline{\mathcal{A}}_i^{(j)}$ is an optimal-scoring completion of $\mathcal{A}_i$ with respect to parameters $x^{(j)}$,

$$f_{x^{(j)}}\Big(\overline{\mathcal{A}}_i^{(j)}\Big) \;\leq\; f_{x^{(j)}}\Big(\overline{\mathcal{A}}_i^{(j-1)}\Big).$$

In other words, under parameters $x^{(j)}$, the new completions $\overline{\mathcal{A}}^{(j)}$ score just as close to optimal as the old completions $\overline{\mathcal{A}}^{(j-1)}$. This implies that with respect to error,

$$E\Big(x^{(j)},\, \overline{\mathcal{A}}^{(j)}\Big) \;\leq\; E\Big(x^{(j)},\, \overline{\mathcal{A}}^{(j-1)}\Big),$$

whether we consider relative or absolute error. So for the new completions $\overline{\mathcal{A}}^{(j)}$, error e_j is achievable, as witnessed by $x^{(j)}$. Since the optimum error for $\overline{\mathcal{A}}^{(j)}$, given by the new parameters $x^{(j+1)}$, cannot be worse,

$$e_{j+1} \;\leq\; e_j.$$

Furthermore, e^* lower bounds the error for all completions. □

Theorem 1 implies that the error of this iterative scheme converges, as the error across iterations forms a nonincreasing sequence that is bounded from below. (The error may converge, however, to a value larger than the optimum error e^*.) As shown in Section 4, choosing a good initial completion can reduce the final error. In practice, we iterate this scheme until the improvement in error becomes too small or a bound on the number of iterations is reached. As error improves across the iterations, recovery of the examples generally improves as well.

3.3 Eliminating Degeneracy

In general, the preceding formulations of inverse alignment, for both relative and absolute error, when applied to global sequence alignment have a family $\mathcal{F}$ of *degenerate solutions*:

$$(\sigma_{ab}, \gamma, \lambda) \;\in\; \mathcal{F} \;:=\; \big\{(2c, 0, c) \;:\; c \geq 0\big\}.$$

Every parameter choice $x \in \mathcal{F}$ from this family makes *all* global alignments of an example's sequences have the *same* score, namely, $c(m+n)$ for two sequences of lengths m and n. Such an x makes each example be an optimal scoring alignment, hence such an x is an optimal solution for every instance of inverse alignment. Of course, these degenerate solutions are not of biological interest. To eliminate them, we use the following approach.

Let *nondegeneracy threshold* τ be the difference between the expected cost of a random substitution (of different letters) and the expected cost of a random identity, measured for a default substitution scoring matrix. For a sound scoring scheme that distinguishes between substitutions and identities, this difference should be positive. The value of τ for the commonly used BLOSUM [17] and PAM [5] matrices for standard amino acid frequencies is around 0.4 when substitution scores are translated and scaled to costs in the range [0, 1]. Values of τ for a wide range of BLOSUM matrices are shown in the next section in Fig. 2.

Given a value for threshold τ, we add to the linear program the *nondegeneracy inequality*

$$\frac{\sum_{a<b}\, p_a p_b\, \sigma_{ab}}{\sum_{a<b}\, p_a p_b} \;-\; \frac{\sum_a\, p_a^2\, \sigma_{aa}}{\sum_a\, p_a^2} \;\geq\; \tau, \tag{5}$$

where in the summations a and b range over all amino acids, and p_a is the probability of amino acid a appearing in a random protein sequence. Note that this is a linear inequality in parameters σ_{ab} and σ_{aa}.

Inequality (5) is equivalent to requiring that for the learned parameters, the difference between the expected score of a random substitution and the expected score of a random identity is at least τ. When τ is positive, which holds for standard scoring schemes, this inequality cuts off the degenerate solution $(\sigma_{ab}, \gamma, \lambda) = (2c, 0, c)$, as the following theorem states. Furthermore, desirable solutions (such as the scoring scheme from which τ was measured) are not eliminated.

Theorem 2 (Eliminating Degeneracy). *When threshold $\tau > 0$, inequality (5) eliminates all degenerate solutions $x \in \mathcal{F}$.*

Proof. The degenerate solutions have $\sigma_{ab} = \sigma_{aa} = 2c$, so their expected substitution and identity scores are equal. The nondegeneracy inequality then reduces to $\tau \leq 0$, contradicting $\tau > 0$. In other words, every solution from $\mathcal{F}$ violates inequality (5). □

In essence, the nondegeneracy inequality forces the substitution and identity scores in the optimal solution x^* to the linear program to be at least as nondegenerate as the default matrix from which τ was measured. In the experiments described in the next section, we use the value of τ corresponding to BLOSUM62, namely, $\tau = 0.4$.

TABLE 1
Data Set Characteristics

Data sets	number of benchmarks	number of sequences	sequence length	percent identity	Core blocks coverage	Core blocks identity
U	102	14	239	29.1	40.4	35.8
P	51	15	239	29.7	41.0	37.1
Q	51	12	239	28.1	39.9	33.8
S	25	20	245	27.4	33.2	33.5

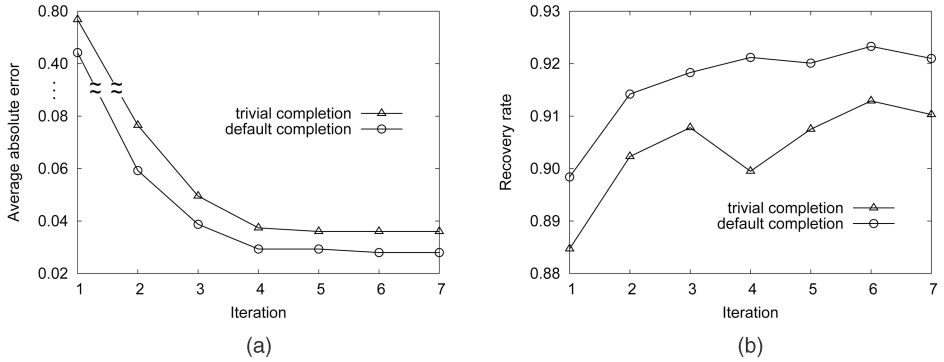

Fig. 1. Improvement in error and recovery for the iterative approach to partial examples. (a) Plot of the error at each successive iteration. (b) Plot of the corresponding recovery for the parameters learned at that iteration. The curves in the plots start from two different initial completions: the trivial completion and the default completion (defined in the text). The examples are all induced pairwise alignments from PALI benchmark b.1.8.1.

We emphasize that the nondegeneracy inequality is crucial. Without it, the inverse alignment algorithm for absolute error immediately outputs the optimal but trivial solution $x = (0, \ldots, 0)$.

4 Experimental Results

To evaluate the performance of this approach to inverse alignment, we ran several types of experiments on biological data. Our main interest is in comparing the new absolute error formulation to the prior relative error formulation, and in investigating whether parameters learned from pairwise alignment examples can improve the recovery of benchmarks when used for multiple sequence alignment. We also examine how the initial completion and the nondegeneracy threshold affect the final recovery.[1]

To obtain the examples for our experiments, we used benchmarks from the PALI [2], [10] suite of structural multiple alignments of proteins. PALI contains a benchmark multiple alignment for every family from the SCOP [25], [26] classification of proteins, computed by structural alignment without hand curation. In total, PALI has 1,655 benchmark alignments, from which we selected a subset U consisting of *all* PALI alignments on at least seven sequences with nontrivial gap structure. (If an alignment had essentially no gaps other than at the terminal ends of its sequences, it was deemed to have *trivial* gap structure and was not included in set U.) Set U was further partitioned into two equal-size subsets P and Q for the purpose of conducting training-set/test-set cross-validation experiments. We also performed a detailed study of a smaller subset $S \subset U$ containing the 25 benchmarks from U with the most sequences. For each benchmark in the smaller set S, where the benchmark is a multiple alignment on n sequences, we took for the examples the $\binom{n}{2}$ pairwise alignments induced on all pairs of rows of the multiple alignment. For the benchmarks in the larger sets U, P, and Q, we took as the examples a much sparser set of induced pairwise alignments, as described later.

Table 1 summarizes the characteristics of these data sets. Each of the PALI benchmarks in these data sets consists of partial examples. The reliable regions in the partial examples are given by the *core blocks* of the benchmark. (PALI explicitly identifies the core blocks in each benchmark, which consist of those columns of the multiple alignment in which all pairs of residues are within 3 Å on their C^α atoms in the structural multiple alignment.) For sets U, P, Q, and S, the table reports the number of benchmarks in each set and, averaged across the benchmarks in the set, their number of sequences, their sequence length, and the percent identity of their induced pairwise alignments. Also shown averaged over the core blocks of the benchmarks are their percent coverage of the sequences and their percent identity.

Fig. 1 illustrates the improvement in error for the iterative approach to partial examples discussed in Section 3.2. The plots show the average absolute error and the recovery rate across the iterations, starting from a given initial completion. The *recovery rate* is the percentage of columns from reliable regions that are present in an optimal alignment computed using the estimated parameters. Results are plotted for two initial completions: the *default completion*, which optimally aligns the unreliable regions using default parameters, and the *trivial completion*, which uses all columns of the partial alignment including unreliable ones.

1. Our experiments do not compare the linear programming approach for inverse alignment to alternative machine learning approaches [7], [37]. The CONTRAlign [6] tool that implements the work of Do et al. [7] does not directly produce a substitution matrix and pair of affine gap penalties for the standard protein alignment scoring model, and reimplementing their approach to facilitate a direct comparison requires nonconvex numerical optimization. The work of Yu et al. [37] is on sequence-to-structure protein alignment, and reimplementing their approach to apply it to standard sequence-to-sequence alignment requires quadratic programming.

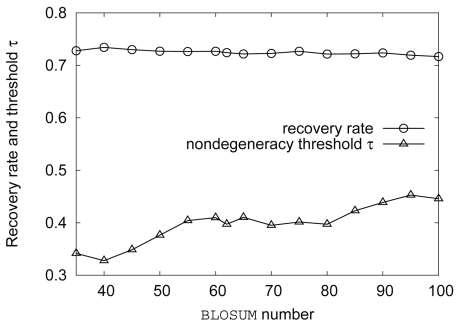

Fig. 2. Robustness of the recovery rate to nondegeneracy threshold τ. The bottom curve plots threshold τ measured on substitution matrix BLOSUMi at each i. The top curve plots the corresponding recovery rate when the value of τ from BLOSUMi is used in the nondegeneracy inequality. Parameters are learned and recovery is measured using the examples in set $\widetilde{U}$ (defined in the text). As shown, recovery is robust to the particular matrix used to establish τ.

(The default parameters are those used by the multiple alignment tool Opal [35], [36], which are the BLOSUM62 substitution matrix with carefully chosen gap penalties.) The examples for the plot are all induced pairwise alignments of PALI benchmark b.1.8.1, and the error criterion is the average absolute error. As the figure shows, the error in alignment scores decreases across the iterations, and the corresponding recovery of the example alignments tends to improve as well. (Also note that starting from the default completion has better recovery and less error than starting from the trivial completion.) Generally, smaller error correlates with higher recovery.

Fig. 2 shows that across a very wide range of conventional substitution matrices, using the corresponding value of threshold τ in nondegeneracy inequality (5) does not adversely affect recovery. In other words, the choice of which substitution matrix is used to establish τ is not critical. (Nevertheless, enforcing the nondegeneracy inequality is crucial: without it, the inverse alignment algorithm immediately outputs an optimal *degenerate* solution.) In the plot, the value i along the horizontal axis specifies the index of a BLOSUM substitution matrix. The BLOSUM matrices we consider have been transformed into cost matrices with extreme values of 0 and 1. The bottom curve plots the value of τ measured on matrix BLOSUMi. (Note that BLOSUMi matrices that correspond to a higher percent identity generally have greater discrimination between substitutions and identities, as evidenced by the bottom curve for τ tending to be increasing in i.) The top curve plots recovery rates when the corresponding value of τ is used in the nondegeneracy inequality. At every point on this curve, parameters are learned using the same collection of pairwise alignment examples (set $\widetilde{U}$, which is described later), and the average recovery rate on these examples is plotted.[2] Across the range of BLOSUM matrices, τ varies up to 40 percent, while there is little variation in recovery. In short, recovery is robust to the particular value τ used in the nondegeneracy inequality.

Table 2 shows a detailed comparison of recovery rates across several different scenarios for inverse alignment. The rows of the table correspond to the PALI benchmarks in set S, which are named by their SCOP identifier. (The rows are sorted in increasing order on column "default (c)," which is described in more detail later.) For each benchmark, the table reports the average recovery rate across its examples, which consist of all induced pairwise alignments of the benchmark, for various scenarios. In these scenarios, parameters learned from the examples for a given benchmark are applied to the sequences in this benchmark, either to compute an optimal global *pairwise alignment* of the example sequences (corresponding to the first group of columns), or to compute a global *multiple alignment* of the benchmark sequences using the tool Opal [35], [36] (corresponding to the second group of columns). Opal is a multiple sequence alignment tool, based on optimally aligning alignments [20], [30], that seeks to optimize the sum-of-pairs objective [3]. In the first group of five columns, the table reports the recovery rates on the examples for optimal pairwise alignments computed using:

1. the *default* parameters in Opal, namely, the BLOSUM62 matrix [17] with carefully chosen affine gap penalties [35],
2. the BLOSUM62 matrix with affine gap penalties learned using the absolute error criterion,
3. the substitution matrix and gap penalties learned using the *relative* error criterion,
4. the substitution matrix and gap penalties learned using *absolute* error, and
5. the *difference* between the recovery rates of the absolute error and default parameters.

In the second group of columns, the table reports the recovery rates of a multiple sequence alignment of the benchmark sequences computed by Opal using:

6. the *default* parameters of Opal,
7. the substitution matrix and penalties learned under *absolute* error, and
8. the *difference* between these two.

In these experiments, the precision of the binary search in the relative error criterion is 0.01 percent. Parameters for alignment are rounded to integers using a scale of 100.

A key conclusion from examining Table 2 is that the *absolute error criterion* substantially outperforms the relative error criterion with respect to recovery of example alignments (seen by comparing the table columns labeled absolute and relative in the pairwise alignment group). In addition, from inspecting the two difference columns in the table (whose maximum entries are marked by a left arrow), parameters learned under absolute error when used for both global pairwise and multiple alignment outperform the default parameters of Opal, which uses the standard BLOSUM62 matrix, by as much as 25 percent in recovery. (This particular matrix had the best recovery among the BLOSUM family.) Also note that the recovery rates of parameters when used for multiple alignment are generally much higher than when used for pairwise alignment. To summarize, by performing inverse alignment from partial examples one can learn parameters for global pairwise or multiple sequence alignment that are tailored to a given protein family and that yield very high recovery.

2. The recovery rate in this plot is for computing pairwise alignments of the example sequences. Later, Table 3 gives the recovery rate when computing multiple alignments of benchmark sequences, which is substantially higher.

TABLE 2
Recovery Rates on Data Set S for Variations of Inverse Alignment

SCOP identifier	Pairwise alignment					Multiple alignment		
	default (a)	BLOSUM62	relative	absolute (b)	difference ($b-a$)	default (c)	absolute (d)	difference ($d-c$)
c.95.1.1	38.5	39.0	34.3	47.8	**9.3**	45.2	70.1	**24.9** ←
d.32.1.3	46.2	45.4	38.9	56.6	**10.4**	64.3	84.7	**20.4**
c.95.1.2	44.1	46.8	32.9	69.4	**25.3**	64.9	82.9	**18.0**
e.3.1.1	46.6	46.2	33.1	50.6	**4.0**	66.6	77.4	**10.8**
d.185.1.1	46.3	47.8	37.4	60.5	**14.2**	68.0	83.1	**15.1**
b.29.1.11	56.5	56.5	64.5	73.4	**16.9**	69.2	90.1	**20.9**
d.54.1.1	51.3	51.9	54.3	67.4	**16.1**	70.4	91.7	**21.3**
c.56.2.1	59.2	61.6	61.5	84.8	**25.6** ←	78.2	90.9	**12.7**
c.1.8.6	64.8	65.8	61.0	72.3	**7.5**	80.6	93.2	**12.6**
b.43.3.1	57.7	58.6	57.2	76.0	**18.3**	82.2	93.4	**11.2**
a.127.1.1	68.9	71.1	47.5	73.1	**4.2**	82.9	89.8	**6.9**
c.67.1.1	60.7	62.2	46.1	66.6	**5.9**	83.6	88.5	**4.9**
d.81.1.1	67.2	69.2	67.8	85.6	**18.4**	85.2	98.4	**13.2**
b.22.1.1	66.6	67.9	75.1	85.0	**18.4**	86.1	91.8	**5.7**
c.1.11.2	68.4	71.4	64.8	81.2	**12.8**	88.4	95.7	**7.3**
b.1.8.1	78.3	79.2	85.9	91.0	**12.7**	89.6	98.6	**9.0**
a.104.1.1	78.8	80.7	69.0	80.0	**1.2**	89.6	90.7	**1.1**
b.43.4.2	59.9	61.4	65.1	82.0	**22.1**	91.6	92.7	**1.1**
a.93.1.1	76.8	81.1	81.0	88.2	**11.4**	94.2	95.2	**1.0**
c.1.7.1	88.9	89.6	88.9	97.1	**8.2**	94.6	99.5	**4.9**
c.1.1.1	91.0	91.3	93.1	97.4	**6.4**	94.8	96.1	**1.3**
b.50.1.2	86.5	87.0	85.3	88.5	**2.0**	94.9	96.5	**1.6**
d.165.1.1	90.2	91.3	92.6	96.8	**6.6**	95.8	98.9	**3.1**
d.162.1.1	86.2	88.4	90.0	96.3	**10.1**	98.0	100.0	**2.0**
e.1.1.1	94.5	96.0	95.3	96.4	**1.9**	98.9	99.1	**0.2**
average	65.7	68.3	64.9	78.6	**12.9**	82.3	91.5	**9.2**

Finally, Table 3 presents recovery results from cross-validation experiments. Parameters learned on sparse training sets using the absolute error criterion are applied to complete test sets. The recovery rate for these parameters is measured on multiple alignments of the benchmarks computed by Opal. In the table, parameters learned on training sets $\widetilde{U}$, $\widetilde{P}$, and $\widetilde{Q}$ using the absolute error criterion are applied to test sets U, P, and Q. For a generic set X of PALI benchmarks, the examples in training set $\widetilde{X}$ are a subset of the induced pairwise alignments of the benchmarks in X. Set $\widetilde{X}$ contains pairwise alignments selected according to their recovery rate in an initial multiple alignment of the benchmark computed by Opal using its default parameters. For $\widetilde{P}$ and $\widetilde{Q}$, the subset contains the induced pairwise alignments from each benchmark whose initial recovery rates rank at the 25th, 50th, and 75th percentiles; for $\widetilde{U}$, the examples rank at the 33rd and 66th percentiles. Parameters from a given training set were used in Opal to compute multiple alignments of the benchmarks in the test set, and the table reports the average recovery.

Note that there is only a small difference in recovery when parameters are applied for multiple sequence alignment to disjoint test sets, compared to their recovery on their training set. This suggests that the absolute error method is not overfitting the parameters to the training data.

To give a sense of running time, performing inverse alignment on training set $\widetilde{U}$ of more than 200 examples involved around 6 iterations for completing partial examples and took about 4 hours total on a 3.2 GHz Pentium 4 processor with 1 Gbyte of RAM, using GLPK [23] to solve the linear programs. An iteration took roughly 40 minutes and required around 4,000 cutting planes.

TABLE 3
Recovery Rates for Cross-Validation Experiments

Training set characteristics			Test set recovery		
data set	examples	identity	U	P	Q
$\widetilde{U}$	204	33.1	83.4	84.8	82.0
$\widetilde{P}$	153	34.5	82.9	86.1	82.2
$\widetilde{Q}$	153	31.9	82.8	84.4	81.2

5 Conclusion

We have explored a new approach to inverse parametric sequence alignment that for the first time carefully treats partial examples. This new approach minimizes the average absolute error of alignment scores, and iterates over completions of partial examples. We also studied for the first time the performance of the resulting scoring schemes when used for multiple sequence alignment. A key conclusion from this study is that the new absolute error criterion outperforms the prior relative error criterion in terms of recovery rate. Our results also indicate that a substantial improvement in alignment accuracy can be achieved on individual protein families, and that scoring schemes learned from a sample of families generalize well to other protein families.

5.1 Further Research

There are many directions for further research. Given that we can solve instances of inverse alignment with hundreds of parameters, is it possible to significantly increase the accuracy of protein sequence alignment through a more sophisticated scoring model with additional parameters that incorporate features such as amino acid *hydrophobicity*

[7], [32], [8] and predicted *secondary structure* [38], [29], [22]? Can accuracy be boosted by modifying inverse alignment to directly incorporate *example recovery* [37] into the formulation? Is parameter generalization benefited by including a *regularization* term in the objective function [7], [37], for instance by penalizing parameter overfitting through the L_1 norm to retain a linear programming formulation? There remains much to investigate.

ACKNOWLEDGMENTS

The authors wish to thank Chuong Do for helpful discussions, and Travis Wheeler for assistance with using `Opal` [36]. We also thank the anonymous referees for suggestions that improved the paper. This research was supported by the US National Science Foundation through Grant DBI-0317498. An earlier version of this paper appeared in [21].

REFERENCES

[1] 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, 2001.

[2] S. Balaji, S. Sujatha, S.S.C. Kumar, and N. Srinivasan, "PALI: A Database of Alignments and Phylogeny of Homologous Protein Structures," *Nucleic Acids Research,* vol. 29, no. 1, pp. 61-65, 2001.

[3] H. Carrillo and D. Lipman, "The Multiple Sequence Alignment Problem in Biology," *SIAM J. Applied Math.,* vol. 48, pp. 1073-1082, 1988.

[4] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver, *Combinatorial Optimization.* John Wiley & Sons, 1998.

[5] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt, "A Model of Evolutionary Change in Proteins," *Atlas of Protein Sequence and Structure,* M.O. Dayhoff, ed., vol. 5, no. 3, pp. 345-352, Nat'l Biomedical Research Foundation, 1978.

[6] C. Do, *CONTRAlign: CONditional TRAining for Protein Sequence Alignment,* version 1.04, http://contra.stanford.edu/contralign/, 2005.

[7] C. Do, S. Gross, and S. Batzoglou, "CONTRAlign: Discriminative Training for Protein Sequence Alignment," *Proc. 10th Conf. Research in Computational Molecular Biology (RECOMB '06),* pp. 160-174, 2006.

[8] R.C. Edgar, "MUSCLE: Multiple Sequence Alignment with High Accuracy and High Throughput," *Nucleic Acids Research,* vol. 32, pp. 1792-1797, 2004.

[9] D. Eppstein, "Setting Parameters by Example," *SIAM J. Computing,* vol. 32, no. 3, pp. 643-653, 2003.

[10] V.S. Gowri, S.B. Pandit, B. Anand, N. Srinivasan, and S. Balaji, *PALI: Phylogeny and ALIgnment of Homologous Protein Structures,* release 2.3, http://pauling.mbu.iisc.ernet.in/~pali, 2005.

[11] J.R. Griggs, P. Hanlon, A.M. Odlyzko, and M.S. Waterman, "On the Number of Alignments of k Sequences," *Graphs and Combinatorics,* vol. 6, pp. 133-146, 1990.

[12] M. Grötschel, L. Lovász, and A. Schrijver, "The Ellipsoid Method and Its Consequences in Combinatorial Optimization," *Combinatorica,* vol. 1, pp. 169-197, 1981.

[13] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization.* Springer, 1988.

[14] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge Univ. Press, 1997.

[15] D. Gusfield, K. Balasubramanian, and D. Naor, "Parametric Optimization of Sequence Alignment," *Algorithmica,* vol. 12, pp. 312-326, 1994.

[16] D. Gusfield and P. Stelling, "Parametric and Inverse-Parametric Sequence Alignment with XPARAL," *Methods in Enzymology,* vol. 266, pp. 481-494, 1996.

[17] S. Henikoff and J.G. Henikoff, "Amino Acid Substitution Matrices from Protein Blocks," *Proc. Nat'l Academy of Sciences of the USA,* vol. 89, pp. 10915-10919, 1992.

[18] R.M. Karp and C.H. Papadimitriou, "On Linear Characterization of Combinatorial Optimization Problems," *SIAM J. Computing,* vol. 11, pp. 620-632, 1982.

[19] J. Kececioglu and E. Kim, "Simple and Fast Inverse Alignment," *Proc. 10th Conf. Research in Computational Molecular Biology (RECOMB '06),* pp. 441-455, 2006.

[20] J. Kececioglu and D. Starrett, "Aligning Alignments Exactly," *Proc. Eighth ACM Conf. Research in Computational Molecular Biology (RECOMB '04),* pp. 85-96, 2004.

[21] E. Kim and J. Kececioglu, "Inverse Sequence Alignment from Partial Examples," *Proc. Seventh EATCS/ISCB Workshop Algorithms in Bioinformatics (WABI '07),* pp. 359-370, 2007.

[22] Y. Lu and S.-H. Sze, "Multiple Sequence Alignment Based on Profile Alignment of Intermediate Sequences," *Proc. 11th Conf. Research in Computational Molecular Biology (RECOMB '07),* pp. 283-295, 2007.

[23] A. Makhorin, *GLPK: GNU Linear Programming Kit,* release 4.8, http://www.gnu.org/software/glpk/, 2005.

[24] K. Mizuguchi, C.M. Deane, T.L. Blundell, and J.P. Overington, "HOMSTRAD: A Database of Protein Structure Alignments for Homologous Families," *Protein Science,* vol. 7, pp. 2469-2471, 1998.

[25] A.G. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia, "SCOP: A Structural Classification of Proteins Database for the Investigation of Sequences and Structures," *J. Molecular Biology,* vol. 247, pp. 536-540, 1995.

[26] A. Murzin, J.-M. Chandonia, A. Andreeva, D. Howorth, L. Lo Conte, B. Ailey, S. Brenner, T. Hubbard, and C. Chothia, *SCOP: Structural Classification of Proteins,* release 1.69, http://scop.berkeley.edu, 2005.

[27] L. Pachter and B. Sturmfels, "Parametric Inference for Biological Sequence Analysis," *Proc. Nat'l Academy of Sciences of the USA,* vol. 101, no. 46, pp. 16138-16143, 2004.

[28] M.W. Padberg and M.R. Rao, "The Russian Method for Linear Programming III: Bounded Integer Programming," Technical Report 81-39, Graduate School of Business and Administration, New York Univ., 1981.

[29] J. Pei and N.V. Grishin, "PROMALS: Towards Accurate Multiple Sequence Alignments of Distantly Related Proteins," *Bioinformatics,* vol. 23, no. 7, pp. 802-808, 2007.

[30] D. Starrett, T.J. Wheeler, and J.D. Kececioglu, *AlignAlign: Software for Optimally Aligning Alignments,* version 0.9.7, http://alignalign.cs.arizona.edu, 2005.

[31] F. Sun, D. Fernández-Baca, and W. Yu, "Inverse Parametric Sequence Alignment," *J. Algorithms,* vol. 53, pp. 36-54, 2004.

[32] J.D. Thompson, D.G. Higgins, and T.J. Gibson, "CLUSTAL W: Improving the Sensitivity of Progressive Multiple Sequence Alignment through Sequence Weighting, Position-Specific Gap Penalties and Weight Matrix Choice," *Nucleic Acids Research,* vol. 22, pp. 4673-4680, 1994.

[33] 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, 2005.

[34] M. Waterman, M. Eggert, and E. Lander, "Parametric Sequence Comparisons," *Proc. Nat'l Academy of Sciences of the USA,* vol. 89, pp. 6090-6093, 1992.

[35] T.J. Wheeler and J.D. Kececioglu, "Multiple Alignment by Aligning Alignments," *Proc. 15th ISCB Conf. Intelligent Systems for Molecular Biology (ISMB '07), Bioinformatics,* vol. 23, pp. i559-i568, 2007.

[36] T.J. Wheeler and J.D. Kececioglu, *Opal: Software for Aligning Multiple Biological Sequences,* version 0.3.7, http://opal.cs.arizona.edu, 2007.

[37] C.-N. Yu, T. Joachims, R. Elber, and J. Pillardy, "Support Vector Training of Protein Alignment Models," *Proc. 11th Conf. Research in Computational Molecular Biology (RECOMB '07),* pp. 253-267, 2007.

[38] H. Zhou and Y. Zhou, "SPEM: Improving Multiple Sequence Alignment with Sequence Profiles and Predicted Secondary Structures," *Bioinformatics,* vol. 21, no. 18, pp. 3615-3621, 2005.

Eagu Kim received the MS degree in computer science from Yonsei University, Korea, in 1998, with a thesis in cryptography. He is a PhD student in the area of algorithms for computational biology in the Department of Computer Science, University of Arizona.

John Kececioglu received the PhD degree in computer science from the University of Arizona in 1991. He did postdoctoral study at the Université de Montréal and the University of California, Davis, then taught at the University of Georgia, before joining the Department of Computer Science, University of Arizona in 2000, where he is an associate professor of computer science. He is the recipient of a US National Science Foundation CAREER Award, has served on the Scientific Advisory Board of the Max-Planck-Institut für Informatik, and serves on the Editorial Board of *Algorithms for Molecular Biology*. His research is in the design, analysis, and implementation of algorithms for computational biology.