Pareto optimal pairwise sequence alignment

Kevin W DeRonne and George Karypis

Abstract—Sequence alignment using evolutionary profiles is a commonly employed tool when investigating a protein. Many profile-profile scoring functions have been developed for use in such alignments, but there has not yet been a comprehensive study of Pareto optimal pairwise alignments for combining multiple such functions. We show that the problem of generating Pareto optimal pairwise alignments has an optimal substructure property, and develop an efficient algorithm for generating Pareto optimal frontiers of pairwise alignments. All possible sets of two, three and four profile scoring functions are used from a pool of eleven functions and applied to 588 pairs of proteins in the ce_ref dataset. The performance of the best objective combinations on ce_ref is also evaluated on an independent set of 913 protein pairs extracted from the BAliBASE RV11 dataset. Our dynamic-programming-based heuristic approach produces approximated Pareto optimal frontiers of pairwise alignments which contain comparable alignments to those on the exact frontier, but on average in less than 1/58th the time in the case of four objectives. Our results show that the Pareto frontiers contain alignments whose quality are better than the alignments obtained by single objectives. However, the task of identifying a single high-quality alignment among those in the Pareto frontier remains challenging.

Index Terms—Pareto, pairwise sequence alignment, optimization

1 Introduction

Profile-based sequence alignments have long been the workhorse for establishing relations between protein sequences, and are used extensively for studying the properties of uncharacterized proteins. An accurate alignment to a template sequence can help in the inference of protein structure and function. Key to alignment methods is the scheme used to score aligned positions. Various approaches have been developed whose individual performance has been extensively compared [1]–[3]. The focus of this paper is on investigating the extent to which improvements in pairwise protein sequence alignments can be attained by combining different profile-profile scoring functions. Though such scoring functions can be easily combined in an *ad hoc* fashion by using a linear combination to derive a "meta" profile-profile scoring function, this study investigates treating the way in which these functions are combined as a multi-objective optimization problem based on Pareto optimality. When optimizing multiple objectives, a Pareto optimal solution is one in which improving the value of any objective requires the degradation of another.

This paper presents a multi-objective pairwise sequence alignment algorithm using the affine gap model. We show that the problem exhibits optimal substructure, and develop a dynamic programming algorithm to find the Pareto optimal frontier. We present optimizations to the overall approach which limit the computational requirements, along with several approaches for selecting a high-quality solution from the Pareto frontier. Results from a comprehensive study involving 588 pairs of proteins, and all possible combinations of size two, three and four objectives from a pool of eleven are presented. The best performing schemes from this study are also evaluated on the challenging RV11 dataset from BAliBASE [4].

To the best of our knowledge, this paper is the largest study of Pareto optimal alignments both in terms of the number of sequences and the number of objectives involved. We present a novel method to approximate a Pareto frontier, and compare it with an existing evolutionary method. For four objectives, on average our optimizations can produce comparable solutions in 1/58th the time required to generate the exact frontier.

Comparing alignments selected from a Pareto optimal frontier with those produced by a single objective or a linear combination of objectives (our baseline), we find that Pareto frontiers frequently contain alignments of higher quality. However, identifying the best alignment on a Pareto frontier is quite challenging, and none of the selection schemes presented here can consistently pick an alignment of higher quality than the baseline. In contrast, for the same sets of objectives, an evolutionary algorithm only rarely generates higher quality alignments than the baseline.

The remainder of this paper is organized as follows. In section 2 we present preliminary information on which the rest of the paper depends. In section 3 we cover previously published related work. Section 4 introduces the notion of Pareto optimal sequence alignment, proving that the problem has the optimal substructure property. Section 5 explains our experimental methodology, while section 6 presents our results. In section 7 we present a brief discussion and conclude.

• *K.W. DeRonne and G. Karypis are with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 USA E-mail: deronne@cs.umn.edu*

2 PRELIMINARY MATERIAL

2.1 Notation and Definitions

Characters in script (*e.g.*, $\mathcal{V}, \mathcal{E}$ will be used to denote sets, characters with an arrow (*e.g.*, $\vec{x}$) will be used to denote vectors, and boldface characters (*e.g.*, $\mathbf{m}$) will be used to denote multi-dimensional matrices. We will use the letters A and B to denote proteins represented by strings of amino acid residues. The ith residue of protein A will be denoted by a_i. A subsequence of protein A starting at residue a_i and ending at residue a_j will be denoted by $A(i:j)$. An amino acid scoring matrix m is a two dimensional matrix with amino acids as both row and column labels. The size of m will be $|\Sigma| \times |\Sigma|$, where Σ is the amino acid alphabet with the addition of a space character, and $|\Sigma|$ equals the number of characters in that alphabet. Each $m_{i,j}$ entry in this matrix represents the score for substituting amino acid a_i with b_j. Such a matrix is referred to as a *position-to-position scoring matrix*. A global alignment between two strings A and B is obtained by inserting enough spaces into either or both strings such that the resulting strings A' and B' are of equal length l; and then establishing a one-to-one mapping between a'_i and b'_i for $\{i = 1 \ldots l\}$. When A' and B' are strings of amino acids, a protein sequence alignment is computed. Spaces may not be aligned against one another, and any maximal consecutive run of spaces constitutes a *gap*. Under an affine gap model, the score for a global protein sequence alignment of length l is given by $\sum_{i=1}^{l} m_{a'_i,b'_i} - n_g \times go - n_s \times ge$, where go is the cost associated with a gap, n_g is the number of gaps, ge is the cost associated with a space, and n_s is the number of spaces. Collectively, go and ge are referred to as *gap penalties*. As the cost associated with spaces is determined outside of m, the scores for mapping any amino acid to a space are set to zero. The optimal sequence alignment problem under the affine gap model is that of finding the alignment that maximizes the alignment score. The alignment scoring function is the *objective function* of this optimization problem, which is parameterized on go, ge and m.

2.2 Efficient Global Sequence Alignment

Given two protein sequences A and B, the global sequence alignment problem under the affine gap model can be solved using dynamic programming in $O(n_1 n_2)$ time, where n_1 and n_2 are the lengths of A and B [5]. The recurrence relations defining the optimal value of an alignment with affine gap weights are

$$\begin{aligned} F_{i,j} &= \max(F_{i-1,j} - ge, V_{i-1,j} - go - ge), \\ E_{i,j} &= \max(E_{i,j-1} - ge, V_{i,j-1} - go - ge), \\ G_{i,j} &= \max(V_{i-1,j-1} + m_{i-1,j-1}, F_{i,j}, E_{i,j}), \\ V_{i,j} &= \max(E_{i,j}, F_{i,j}, G_{i,j}), \end{aligned} \quad (1)$$

where $F_{i,j}$, $E_{i,j}$, $G_{i,j}$ and $V_{i,j}$ are the the scores of the optimal alignment of the ith prefix of A and the jth prefix of B, without constraint for $V_{i,j}$, but under the constraint that i is aligned against a gap for $F_{i,j}$, j is aligned against a gap for $E_{i,j}$, and i is aligned against j for $G_{i,j}$. (The reader is directed to [6] for a more detailed discussion.)

2.3 Pareto Optimality

Optimizing the value of a single function entails finding either a minimum or maximum value over the range of the function. When optimizing multiple functions simultaneously, one must take into account the values for all of the functions. Consider a set of feasible solutions $\{s_1, \ldots, s_n\}$ to an optimization problem, and let f_1 and f_2 be two objective functions. A solution s_i dominates another solution s_j if $f_1(s_i) \geq f_1(s_j)$ and $f_2(s_i) > f_2(s_j)$, or if $f_1(s_i) > f_1(s_j)$ and $f_2(s_i) \geq f_2(s_j)$. If $f_1(s_i) \geq f_1(s_j)$, and $f_2(s_i) \leq f_2(s_j)$ or $f_1(s_i) \leq f_1(s_j)$, and $f_2(s_i) \geq f_2(s_j)$, then neither set dominates the other. We will use the notation $f_1 \succ f_2$ to indicate that f_1 dominates f_2. Over all feasible solutions optimizing f_1 and f_2, we can construct a set of values such that no pair strictly dominates another pair in that set. This set is known as the Pareto frontier, and the points are referred to as being *Pareto optimal*. With respect to optimizing f_1 and f_2, each point in this set is considered equivalent to all the other points, and no other points need to be considered when trying to optimize the functions involved.

3 RELATED RESEARCH

There has been a fairly limited amount of work done on multi-objective sequence alignment, with notable exceptions occurring within the context of RNA secondary structure prediction [7], [8], multiple sequence alignment [9], [10] and treating gap penalties as an objective to be optimized [11]–[13]. In terms of the methods used to perform multi-objective alignments, the work done with RNA and multiple sequence alignment has relied upon evolutionary algorithms, while the work done with gap penalties as an objective has used dynamic programming. Evolutionary algorithms attempt to optimize an objective function by mimicking the process of natural selection. A considerable amount of effort has been devoted to applying these algorithms to multi-objective optimization problems [14]–[18], and also to perform sequence alignments using a single objective [19]–[22] or multiple objectives [7]–[10], [23]. An evolutionary algorithm consists of three primary parts: an initial population of solutions, a set of genetic operators which modify those solutions, and an objective function to assess the quality (or *fitness*) of those solutions. By applying the genetic operators to the initial population, new solutions are generated and fitness is determined. Then a new population is selected from the new solutions, the previous solutions, or a combination of the two. This process is repeated until some predetermined stopping criteria are met. For the single objective alignment applications, the fitness assessment is straightforward, but for multiple objectives the process

is more complicated. Two approaches for this assessment are random tournaments between solutions in which one solution "wins" and is considered to have higher fitness [10] and using dominance-rank assignment [7]–[9]. Dominance-rank assignment consists of organizing a set of points into ordered Pareto optimal frontiers. Points on the rank one frontier are not dominated by any points. Each point on the rank two frontier is dominated by some point on the rank one frontier, and so on, with all points on the rank r frontier being dominated by some point on each frontier with rank $< r$. As ranking solutions is computationally expensive, several algorithms for this have been reported [24]–[26].

Previous dynamic programming-based approaches have focused on treating gap penalties, rather than position-to-position scores, as objective functions. In parametric sequence alignment [12], the space defined by the gap penalties is partitioned such that the resulting regions are as large as possible, and that one alignment is optimal for each region. Given a set of sequences and a range of values for gap penalties, [12] finds all optimal alignments within that range. In an attempt to do away with gap penalties entirely, [11] and [13] use as their objective functions counts of spaces and matches (positions in an alignment where both sequences have the same symbol). These algorithms attempt to maximize the number of matches while minimizing the number of spaces in the alignment.

4 Pareto Optimal Sequence Alignment

This paper focuses on generating optimal multi-objective sequence alignments; specifically alignments whose scores for each objective correspond to points on a Pareto frontier. We refer to such alignments as *Pareto optimal alignments*. Formally, the Pareto optimal alignment problem is defined as follows:

Definition. Given a pair of sequences A and B, and a set of k alignment scoring functions (objectives) $\{f_1, \ldots, f_k\}$, the Pareto optimal alignment problem is that of generating the set of alignments that constitute the complete Pareto frontier.

This problem has the property of optimal substructure, which is defined as follows:

Lemma. Given an alignment (A', B') of sequences A and B on the Pareto frontier, and a pair of indices i, j with $i < j$ such that i and j are not cutting through a gap, then the alignment $(A'[i : j], B'[i : j])$ is also an alignment on the Pareto frontier for the substrings of A and B in $A'[i : j]$ and $B'[i : j]$.

The correctness of this lemma can be shown by contradiction. Let A'' and B'' be the strings of A and B, respectively, that fall within the $(A'[i : j], B'[i : j])$ portion of the alignment. Assume that $(A'[i : j], B'[i : j])$ is not on the Pareto frontier of A'' and B''. This means that there is another alignment of A'' and B'', call it (A''', B''') whose scores dominate $(A'[i : j], B'[i : j])$. Consider now the alignment of length e of A and B that is obtained by replacing $(A'[i : j], B'[i : j])$ in (A', B') with (A''', B'''). Now let (l, c, r) be the multi-objective vector scores of the $(A'[1 : i-1], B'[1 : i-1])$, $(A'[i : j], B'[i : j])$ and $(A'[j+1 : e], B'[j+1 : e])$ parts of the alignment and c' be the multi-objective score vector of the (A''', B''') alignment. Note that $l + c + r$ is the multi-objective score of (A', B') and $l + c' + r$ is the multi-objective score of the new alignment. Since $c' \succ c$, then $l + c' + r \succ l + c + r$. Hence the new alignment dominates (A', B'); This is a contradiction, as (A', B') is an alignment in the Pareto frontier of A and B. Thus, $(A'[i : j], B'[i : j])$ must belong to the Pareto frontier of global alignments for A'' and B''.

The optimal substructure property of the Pareto optimal alignment problem allows us to develop a dynamic programming algorithm for solving it. The set of recurrence relations associated with this algorithm are similar in nature to those of Equation 1, but these need to be extended to deal with the multiple alignments that exist on the Pareto frontier.

In this extended setting, each entry in the V, F, G and E matrices must store a set of Pareto optimal alignments for the ith and jth prefixes of the two sequences. To represent the change from matrices containing values to matrices containing sets, we replace V, F, G and E with $\mathcal{V}, \mathcal{F}, \mathcal{G}$ and $\mathcal{E}$, respectively. Additionally, instead of single values for go and ge we now have objective-specific gap penalties, so we replace go and ge with the vectors $\vec{go}$ and $\vec{ge}$. Each entry in $\vec{go}$ and $\vec{ge}$ contains a gap penalty for a specific objective. Finally, we have a separate position-to-position scoring matrix for each objective. To represent this change, we replace m with $\mathbf{m}$. The $\mathbf{m}$ parameter is a three-dimensional matrix with amino acids as labels for the first two dimensions, and objectives as labels for the third. For k objectives, the size of $\mathbf{m}$ will be $|\Sigma| \times |\Sigma| \times k$, where Σ is the amino acid alphabet with the addition of a space character, and $|\Sigma|$ equals the number of characters in that alphabet. An entry $\vec{\mathbf{m}}_{a_i,b_j}$ constitutes a vector of k scores, one for substituting amino acid a_i with b_j under each of the k objectives.

For each pair i, j in $\mathcal{V}, \mathcal{F}, \mathcal{G}$ and $\mathcal{E}$, we have a set of partial alignments (solutions) from the beginning of each sequence to i, j, based on applying the original recurrence relations to each objective using the $\vec{go}$, $\vec{ge}$ and $\mathbf{m}$ scoring parameters. These solution sets will consist of vectors of Pareto optimal objective scores resulting from combinations of Pareto frontiers at previous positions in the alignment. For example, $\mathcal{F}_{i,j}$ will consist of combining solutions on the frontiers stored at $\mathcal{F}_{i-1,j}$ and $\mathcal{V}_{i-1,j}$. However, when taken together these solutions may not all be Pareto optimal, so we eliminate dominated solutions from the set for i, j and store only the Pareto frontier. This operation, PF, replaces the $\max$ function in the original recurrence relations. Given a set of solutions, the PF operator generates a subset of solutions that

comprise a Pareto optimal frontier. Additionally, we define an operator $\{\}$ which adds or subtracts a given vector $\vec{x}$ from each member in a set of vectors $\mathcal{Y}$:

$$\{\mathcal{Y} - \vec{x}\} \rightarrow \forall \vec{y} \in \mathcal{Y} : \vec{y} - \vec{x}.$$

The recurrence relations for Pareto optimal sequence alignment, then, are

$$\begin{aligned}
\mathcal{F}_{i,j} &= \text{PF}(\{\mathcal{F}_{i-1,j} - \vec{ge}\} \cup \{\mathcal{V}_{i-1,j} - \vec{go} - \vec{ge}\}),\\
\mathcal{E}_{i,j} &= \text{PF}(\{\mathcal{E}_{i,j-1} - \vec{ge}\} \cup \{\mathcal{V}_{i,j-1} - \vec{go} - \vec{ge}\}),\\
\mathcal{G}_{i,j} &= \text{PF}(\{\mathcal{V}_{i-1,j-1} + \vec{\mathbf{m}}_{i-1,j-1}\} \cup \mathcal{F}_{i,j} \cup \mathcal{E}_{i,j}\}),\\
\mathcal{V}_{i,j} &= \text{PF}(\mathcal{E}_{i,j} \cup \mathcal{F}_{i,j} \cup \mathcal{G}_{i,j}), \qquad (2)
\end{aligned}$$

where $\mathcal{F}_{i,j}$, $\mathcal{E}_{i,j}$, $\mathcal{G}_{i,j}$ and $\mathcal{V}_{i,j}$ are the sets of scores for the Pareto optimal alignments between the ith prefix of A and the jth prefix of B, without constraint for $\mathcal{V}_{i,j}$, but under the constraint that i is aligned against a gap for alignments in $\mathcal{F}_{i,j}$, j is aligned against a gap for alignments in $\mathcal{E}_{i,j}$, and i is aligned against j for alignments in $\mathcal{G}_{i,j}$.

4.1 Elimination of Dominated Solutions

Determining which solutions are dominated and which should be kept (*i.e.*, performing the PF operation) can be computationally expensive. A simple approach to finding a Pareto frontier from a set of solutions steps through the list of possible values, eliminating any solution dominated by another solution. This requires $O(kn^2)$ time based on the total number of solutions n and number of objectives k. In the next two sections, we describe two techniques for eliminating dominated solutions. The first of these techniques uses multi-key sorting and the second uses a tree structure.

4.1.1 Sorting-based Elimination

Our algorithm for eliminating dominated solutions treats the case of two objectives differently than when there are more than two. For two objectives, our algorithm proceeds as follows: The solutions are sorted in decreasing order based on the first objective (call this set S_1), and in increasing order based on the second objective (call this set S_2). The first solution in S_1 (call it s_{1_1}) is saved, and S_2 is scanned until reaching s_{1_1}. All solutions seen in S_2 before s_{1_1} are dominated by s_{1_1} and can be eliminated. The process is repeated for the next solution in S_1 until all solutions in S_1 have been examined. As it is dominated by the time required for sorting, this approach has a computational complexity of $O(n \log n)$.

When there are three or more objectives, we proceed as follows. Using a multi-key sort, a list of solutions is sorted by the first objective function, then within equivalent values of the first objective by the second objective function, and so on for all functions. The first solution in this list is definitely on the frontier so it is added to the frontier set. Then the second solution in the list is compared with the first. If the first solution dominates the second, the second solution is discarded. Otherwise, the second solution is added to the frontier set. Continuing down the list, each solution is compared with the entire frontier yet seen, until all solutions have been examined. The complexity of this technique depends on the size of the frontier in question. If the number of solutions on the frontier is relatively small, the complexity is dominated by the sorting portion: $O(kn \log(n))$ where n is the number of solutions. If the number of solutions on the frontier is nearly the same as the initial number of solutions, then the complexity is $O(n^2)$. We refer to this technique as the *Sort-all* method.

In the context of a dynamic programming alignment, we can greatly increase the efficiency of the sorting-based algorithm by leveraging the structure of the problem. The inputs to the elimination algorithm consist of two or three internally optimal Pareto frontiers of solutions (Equation 2). For example, computing the set $\mathcal{F}_{i,j}$ involves combining frontiers $\mathcal{F}_{i-1,j}$ and $\mathcal{V}_{i-1,j}$, after computing $\{\mathcal{V}_{i-1,j} - \vec{go}\}$. Adding or subtracting a constant from all values on a Pareto optimal frontier results in another Pareto optimal frontier, shifting the original points but preserving their relative positions. By definition, no point within a Pareto optimal set can dominate another, and $\mathcal{F}_{i-1,j}$ and $\{\mathcal{V}_{i-1,j} - \vec{go}\}$ are Pareto optimal sets. Thus, when combining these sets, solutions within a set need not be compared with each other. For t frontiers containing n solutions, in the average case, this reduces the cost of eliminating dominated solutions by $O(n^2/t)$. We refer to this technique as the *Merge-front* method.

In an attempt to further decrease the required computational time, we examine eliminating solutions from consideration based on the structure of the dynamic programming matrices. When generating a set of solutions, two input solutions may have the same source but arrive via different routes. For example, consider a solution $\vec{s}$ in the set $\mathcal{V}_{i-1,j-1}$. This solution can appear in $\mathcal{V}_{i,j}$ via three routes: directly from $\mathcal{V}_{i-1,j-1}$, through $\mathcal{F}_{i-1,j}$, and through $\mathcal{E}_{i,j-1}$. In the first route, $\vec{s}$ will have $\vec{\mathbf{m}}_{i-1,j-1}$ added to its scores. In either of the two remaining routes, $\vec{s}$ will have $-\vec{go} - \vec{ge}$ added to its scores before being stored in the intermediate set, then $-\vec{ge}$ added to its scores to reach $\mathcal{V}_{i,j}$. Note that $\vec{s}$ might be eliminated before this point if it is not Pareto optimal for the intermediate set. If $\vec{s}$ does reach $\mathcal{V}_{i,j}$ it will have scores of $\vec{s} + \vec{\mathbf{m}}_{i-1,j-1}$ or $\vec{s} - (\vec{go} + 2\vec{ge})$. Given this, if $\vec{\mathbf{m}}_{i-1,j-1} \geq -(\vec{go} + 2\vec{ge})$ then we can eliminate all solutions arriving from the latter route without comparing them to anything. (The third route is analogous to the second in that it results in the same scores, but passes through $\mathcal{E}_{i,j-1}$ instead of $\mathcal{F}_{i-1,j}$.) We refer to the technique using this optimization as the *Merge-front-A* method.

Another optimization can be made when either $\mathcal{E}$ or $\mathcal{F}$ is the sole contributor to a set in $\mathcal{V}$. For example, when creating the set for $\mathcal{V}_{i,j-1}$, if only solutions from $\mathcal{E}_{i,j-1}$ are used then when constructing $\mathcal{E}_{i,j}$, anything from $\mathcal{V}_{i,j-1}$ can be safely ignored. To see this, recall

that $\mathcal{E}_{i,j} = \text{PF}(\{\mathcal{E}_{i,j-1} - \vec{ge}\} \cup \{\mathcal{V}_{i,j-1} - \vec{go} - \vec{ge}\})$, so if $\mathcal{E}_{i,j-1} = \mathcal{V}_{i,j-1}$, no element of $\{\mathcal{V}_{i,j-1} - \vec{go}\}$ will dominate $\mathcal{E}_{i,j-1}$. We refer to the technique using this optimization as the *Merge-front-B* method.

4.1.2 Dominance Trees

The Dominance tree method described in [25] is also used to identify dominated solutions. This technique eliminates redundant comparisons by maintaining previously computed dominance relationships in a tree structure. Nodes at a given level of the tree dominate all nodes on levels below them, but do not dominate each other. After constructing a dominance tree, we eliminate anything found below the first level of the tree. This is the same method used to rank solutions in the evolutionary algorithm described in [7] and Section 5.6.2.

4.2 Frontier Size Reduction Schemes

The number of solutions produced in the course of constructing a Pareto optimal frontier of alignments can be quite large. At each entry for $\mathcal{V}$, $\mathcal{E}$, $\mathcal{F}$, and $\mathcal{G}$ from Equation 2, the entire frontier needs to be calculated and persisted, which leads to severe computational and storage requirements. In order to keep the problem computationally tractable, we reduce the number of Pareto optimal solutions maintained for each (i, j) sub-problem. Note that we only perform this reduction after dominated solutions have been eliminated by the methods described above, and we have generated a minimum number of solutions (arbitrarily set to 100). We call this process *coarsening* the Pareto frontier.

The goal of coarsening is to eliminate as many solutions as possible while preserving the diversity of the solutions on a Pareto frontier. Put another way, we would like to eliminate only those solutions which are very similar to some solution that is kept. This improves the chances that the Pareto frontier for the complete sequences will contain high-quality alignments. One simple way of achieving this is by randomly selecting a percentage cp of solutions to keep, which should on average select a diverse set. We refer to this technique as the *Sample* method. Varying cp controls how many solutions the Sample method will keep (and thus how many it will eliminate), and is referred to as its *coarsening parameter*.

Randomly sampling the frontier risks keeping multiple solutions which are very similar, so we designed an algorithm which lays a grid over a normalized space of solutions (see Section 4.3.1), and keeps at most one solution from each cell in the grid. The grid is constructed as follows. The possible values for each objective get divided into a fixed number of cells c, so given k objectives there will be c^k possible cells, though not all of them need to be occupied. The width of the cells along an objective v is calculated as $(\max_v - \min_v)/c$ where $\min_v$ is the minimum value seen on the frontier for objective v, and $\max_v$ is the maximum value seen on the frontier for objective v. This means that for a given cell, the size along one objective can be very small while the size along another objective can be quite large. We refer to this technique as the *Cell* method, with c as its coarsening parameter.

If the solutions selected using the Cell method lie close to the borders of their cells, the diversity of the resulting set can be compromised. To address this issue, we use another method which visits the frontier solutions in arbitrary order, eliminating any solutions within a specified Euclidean distance cd of the solution in question, then proceeding to the next solution not previously eliminated and repeating the process. We refer to this technique as the *Centroid* method, with cd as its coarsening parameter.

4.3 Solution Selection

Having generated a Pareto optimal frontier consisting of alignments between a pair of sequences, the problem becomes one of choosing the best possible alignment from the set. We examine three methods for accomplishing this, called the Unit-space method, the Unit-z method, and the Objective Performance Weighted method. Several variations on these techniques were also tested, but with no significant improvement in performance.

4.3.1 Unit-space Selection Method

Taking the maximum values seen for each objective as a single point gives a hypothetical upper-bound for a solution within the context of a pair of sequences. Assuming all objectives are of equal quality, the best solution on the frontier will be the one with the most similar objective scores to this point. The *Unit-space* selection technique chooses the alignment with the smallest Euclidean distance to this point. However, before a meaningful distance can be calculated, all objective scores must be normalized to be in the same range. To achieve this, we create a unit-space normalization of a Pareto frontier as follows. First, if the minimum value seen for objective v ($\min_v$) is negative, we translate all points into the first quadrant by adding $-\min_v$ to each score. Second, we divide each value x_v for objective v by $\max_v$, the maximum value seen for objective v scaled by $-\min_k$, *i.e.*, $x_v/(\max_v - \min_v)$. This scales all values to be in the range $[0, 1]$.

4.3.2 Unit-z Selection Method

Given a distribution of different alignments for a pair of proteins, most of the alignments will be of poor quality. Thus, an alignment which stands out from the background should be a good alignment. A z-score for a point measures how different that point is from a background distribution. It is calculated by subtracting the mean and dividing by the standard deviation for the distribution. By repeatedly shuffling a pair of sequences and aligning the results, we generate a set of objective scores for sequences with the same composition as our input

TABLE 1
Notation used in profile scoring function definitions.

Symbol	Description	Definition
$x \bullet y$	Dot product of x and y	$\sum_a x_a y_a$
Avg(x,y)	Averaged vector	$(x_a + y_a)/2$
$D^{KL}(x,y)$	Kullback-Leibler divergence	$\sum_a x_a \log_2 x_a/y_a$
$H^S(x,y)$	Symmetrized entropy	$(D^{KL}(x,y) + D^{KL}(y,x))/2$
$D^{JS}(x,y)$	Jensen-Shannon divergence	$(D^{KL}(x,\text{Avg}(x,y)) + D^{KL}(y,\text{Avg}(x,y)))/2$
$< x >$	Mean of x	$\sum_a x_a/\|x\|$
$R(x,y)$	Pearson correlation	$(\sum_a x_a - < x > \times \sum_a y_a - < y >)/$ $\sqrt{[\sum_a (x_a^2 - < x >^2) \times \sum_a (y_a^2 - < y >^2)]}$
$\sigma_a(x)$	Rank of x_a in vector x	$\sigma(x) = 1$ if x_a is smallest to $\sigma_a(x) = \|x\|$ if x_a is largest; ties defined so that $\sum_a \sigma_a(x)$ remains constant.

sequences. These scores then serve as the background distribution used to calculate z-scores for all points on the Pareto frontier, which are used in place of the original objective scores. For the *Unit-z* selection method, we take the maximum z-score seen for each objective and combine them to form an upper bound. The solution with objective scores closest to this hypothetical point (in terms of Euclidean distance) is used as the Unit-z selection.

4.3.3 Objective Performance Weighted Method

Both the Unit-z and Unit-space selection methods rely on the assumption that all objectives are of equal quality. In practice this is generally not the case, so to overcome this limitation we assign different weights to each objective, in an attempt to give more emphasis to better objectives. These weights are set to the average match score (see Section 5.4) over all proteins in the ce_ref [27] dataset, then normalized so that they sum to one. The *Objective Performance Weighted Method* chooses a solution as follows. If x_v is the normalized value of objective v, w_{x_v} is the weight for objective v, and k is the number of objectives, then we choose the solution that minimizes $\sqrt{\sum_k w_{x_v}(1 - x_v)^2}$. Note that when $\forall w_{x_v} : w_{x_v} = 1$, this is equivalent to the Unit-space method.

5 Experimental Methodology

5.1 Data

To test the effectiveness of various objective combinations for Pareto optimal sequence alignments, we use the ce_ref set of proteins [27]. This dataset consists of 588 pairs of proteins having high structural similarity but low sequence identity ($\leq$ 30%). However, the supplied alignments are local alignments, and lack a reference point in the global sequences. As such, we cannot directly use them for a meaningful evaluation, so we construct new structural alignments with MUSTANG [28] and use these as our gold standard. (Note that this also makes our results not directly comparable to [3].) Comparing the ce_ref alignments with our MUSTANG-based alignments, we see that over 81% of the amino acid pairs in the local alignments appear in the global alignments.

To further explore the relative performance between single objectives, linear combinations of objectives, combinations of objectives using the evolutionary algorithm and Pareto optimal combinations, we apply these methods to the RV11 dataset from BAliBASE [4]. RV11 is one of the most informative datasets [29] in BAliBASE. This dataset contains 38 multiple sequence alignments, each containing seven or more highly divergent sequences (<20% sequence identity). We exclude one of these multiple sequence alignments, following [30]. From the remaining 37 multiple sequence alignments we extract 921 pairwise alignments. Eight of these pairwise alignments are between protein pairs which are also aligned in the ce_ref set, so we exclude these to create an entirely independent set of 913 pairwise alignments.

5.2 Profile Construction

Many objective functions require profiles as input, and these are generated using PSI-BLAST version 2.2.5 with the options *-h 5 -e 0.1* (these are the same as specified in [3]). PSI-BLAST generates a profile for a query sequence by constructing a multiple sequence alignment between the query and multiple database sequences. This alignment is constrained to be the same length as the query, and is converted into a profile by examining the composition of the amino acids aligned at each position of the query. This profile is then used to search the database in place of the query, and the process is repeated for a specified number of iterations. Preliminary experiments did not show a substantial difference in performance between profile construction methods. The final PSI-BLAST profiles are used as input to most objective functions directly, and to YASSPP for generation of secondary structure predictions. These predictions take the form of a three-dimensional vector, whose values correspond to the prediction scores for each of the three secondary structure states (β-sheet, α-helix and coil), and serve as the basis of the *SS_dotp* objective function.

5.3 Objective Functions

We include eleven objective functions in our study. Nine objectives correspond to the profile-based objective functions from Edgar and Sjölander [3], while the remaining

two are the Picasso [31] objective, and SS_dotp, an objective based on the YASSPP [32] secondary structure prediction vector. Including the secondary structure function adds an interesting dimension, as it is the only one calculated using information from a window of amino acids around the amino acid being scored. Definitions used in the formulas for these functions are listed in Table 1, with the formulas themselves located in Table 2.

5.4 Performance Assessment Metrics

To measure the quality of an alignment, we use the percent of aligned pairs of positions in the test alignment that exist in the gold standard alignment, which we refer to as the *match score* (MS). We find this to be a simple and intuitive method that correlates very well (Pearson correlation 0.98 in our experiments) with the much more complicated Cline-shift method [33]. To aggregate match scores over multiple alignments we present the average match score. In addition, to get a more accurate picture of the relative performance of the alignment selection schemes (including selecting the best alignment on a frontier), we use a metric which we call FIM. FIM, which stands for Fraction IMproved, represents the fraction of alignments in which the best single objective was improved upon. Specifically, using the ranking of single objectives from Table 2, we select the best performing single objective from each combination of objectives to be used for comparison. When making comparisons between multi-objective and single objective approaches, we will refer to this single objective as the *corresponding* single objective. For every alignment, we determine if the multi-objective approach produced a better match score than the corresponding single objective. Counting the number of times this is the case and dividing by the total number of alignments yields the FIM value.

Two metrics are used to compare approximated Pareto frontiers with complete Pareto frontiers. First, the *frontier coverage* metric is used to measure how much of the exact frontier is contained in the approximated frontier. The frontier coverage is calculated as the number of solutions on the exact frontier that are also on the approximated frontier, divided by the total number of solutions on the exact frontier. Second, *percent false positives* is used to measure how many incorrect solutions exist on the approximated frontier. The percent false positives value is calculated as the number of solutions on the approximated frontier that are not on the exact frontier, divided by the total number of solutions on the approximated frontier. A perfect technique would produce a frontier with a frontier coverage of 100% and 0% false positives. Lastly, to compute the statistical significance of our results, we use the student's t-test.

5.5 Gap Parameter Optimization

Accurate alignments require proper gap-open and gap-extension penalties, which must be conditioned both on the objectives used and on alignment type (*e.g.*, global or local). Changing the gap-open and gap-extension values can lead to considerably different alignments, so we take care to set them appropriately. We establish a grid of 400 points (20 on each side), with each point corresponding to a pair of (gap-open, gap-extension) values. Given that a gap will take the place of an alignment between two amino acids, we assume that no gap should cost more than the best aligned pair of amino acids seen. Thus, for a given profile-profile scoring function, the range of values is bounded by $[0, objmax]$, where $objmax$ is empirically determined as the maximum value seen in any position-to-position scoring matrix for 100 protein pairs. Using these values we assign to each grid point the average match score over alignments between the same 100 protein pairs. Areas surrounding local maxima are expanded to form a new grid and the process is repeated until the gain in average match score between iterations drops below 0.001. This is similar to the procedure described in [3] except that we use a larger grid and perform global alignments.

TABLE 2
Average match score, gap parameters and definitions of objective functions.

Name	Go	Ge	Score	Definition
Picasso	9.61	1.92	0.728	$f_1 \bullet p_2 + f_2 \bullet p_1$
Correlf	1.30	0.10	0.725	$R(f_1, f_2)$
Rankp	0.78	0.11	0.712	$R(\sigma(p_1), \sigma(p_2))$
Rankf	0.52	0.08	0.711	$R(\sigma(f_1), \sigma(f_2))$
Ylf	0.27	0.07	0.702	$(1 - D^{JS}(p_1, p_2)) \times (1 + D^{JS}(\text{Avg}(p_1, p_2), p^0)$
Fdotf	0.28	0.01	0.701	$f_1 \bullet f_2$
Yldf	0.27	0.03	0.700	$1 - D^{JS}(f_1, f_2)$
Correlp	1.20	0.07	0.693	$R(f_1, f_2)$
Fdotp	0.20	0.03	0.656	$f_1 \bullet p_2$
Ref	1.50	0.40	0.656	$H^S(f_1, f_2)$
SS_dotp	2.11	0.09	0.585	$ss_1 \bullet ss_2$

Abbreviations used in this table: Go: Gap open cost Ge: Gap extension cost. The Score column shows average match scores over all proteins in the ce_ref set. See Table 1 for more information on objective definitions.

TABLE 3
Runtime performance of methods for the PF operation on four objectives.

Method	Mean runtime	Standard deviation
Sort-all	31.335	70.128
Merge-front	20.657	46.547
Merge-front-A	23.567	52.255
Merge-front-B	20.263	45.995
Dominance-tree	51.299	106.722

5.6 Comparison Algorithms

5.6.1 Linear Combinations

Aside from Pareto optimal combinations, linear combinations are another means of utilizing multiple objectives. We compare our Pareto techniques with a weighted linear combination of objectives. The range of values for each objective is scaled to [0,1] by dividing by the maximum value seen for each objective, and the weights are scaled so that they sum to 1. Objective

TABLE 4
Various performance metrics comparing the best solution on the Pareto frontier with the solution of the best corresponding single objective.

Objectives	Best	Mean	FIM	BPO	Objectives	Best	Mean	FIM	BPO	Objectives	Best	Mean	FIM	BPO
SS Pc	0.761	0.687	0.801	0.728	SS Pc Cp	0.770	0.690	0.855	0.728	SS Pc Cp Rp	0.772	0.691	0.867	0.728
SS Cf	0.759	0.686	0.803	0.726	SS Pc Rp	0.767	0.691	0.861	0.728	SS Pc Cp Rf	0.771	0.693	0.871	0.728
Pc Cp	0.756	0.719	0.757	0.728	SS Pc Cf	0.767	0.691	0.862	0.728	SS Pc Cf Cp	0.771	0.692	0.855	0.728
Pc Rp	0.753	0.720	0.759	0.728	SS Cp Rf	0.767	0.690	0.891	0.712	SS Pc Cf Rp	0.771	0.692	0.874	0.728
Cf Cp	0.751	0.715	0.733	0.726	SS Cf Cp	0.766	0.690	0.854	0.726	SS Cf Cp Rp	0.771	0.690	0.866	0.726

Abbreviations for objectives: SS: SS_dotp Pc: Picasso Cf: Correlf Cp: Correlp Rf: Rankf Rp: Rankp
Description of columns: Best: the MS of the best solution on the Pareto frontier. Mean: the mean MS of solutions on each Pareto frontier individually.
BPO: Best performing objective. FIM: Fraction improved. Values other than FIM are average match scores over 588 protein pairs

values are scaled on a per-protein basis (meaning each position-to-position matrix m is scaled independently), then weights are applied to form a new m matrix for Equation 1. To optimize the weights on the objectives we perform a grid search within the range [0,1], in 0.1 increments. Gap parameters are optimized for each set of weights in the grid (see Section 5.5). For a given set of objectives, the weights and gap parameters with the highest average match score are used. The optimized weights and gap parameters are listed in Table 1 of the supplementary material.

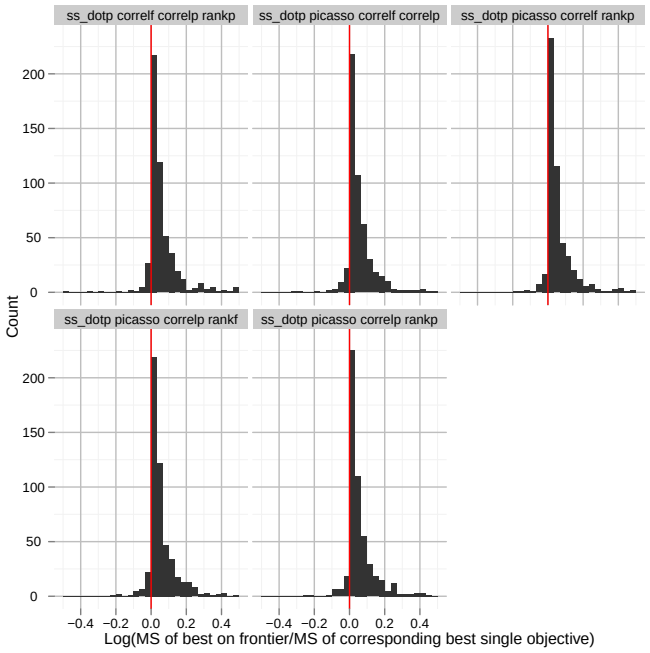
5.6.2 Evolutionary Algorithm

Current multi-objective alignment algorithms typically involve using evolutionary algorithms to approximate the Pareto frontier. Here, we implement the algorithm used in *Cofolga2mo* [7], though we replace the $O(N^3)$ original dominance-rank assignment algorithm with a considerably faster algorithm ($O(N^2)$ in the worst case) [25]. This is the fastest known algorithm that can be used for this problem.

Cofolga2mo begins by initializing a population of alignments using weighted stochastic backtracking [8]. This technique generates a dynamic programming matrix for an alignment, then generates a set of alignments through a nondeterministic backtracking process. The initial population also includes the optimal alignments for each individual objective. To give this algorithm the best chance at covering the actual Pareto frontier, we determine the initial population size as 15 times the number of solutions on the frontier generated using our Pareto method.

After the initialization step, *Cofolga2mo* alternates between two phases, evaluation and reproduction. In the evaluation phase, dominance ranks are assigned to all alignments in the current population. In the reproduction phase, candidates for reproduction are selected with probabilities inversely proportional to their ranks. All the best (rank one) solutions are preserved for the next generation. To create a child solution a genetic operator is randomly selected, then one or two parents are selected based on the needs of the operator. The genetic operator is applied, and if the resulting alignment is valid it replaces one of the parents. (A valid alignment contains no gaps aligned with other gaps.) Five genetic operators are employed: random two-point crossover, random gap-block shuffling, local re-alignment with weighted stochastic backtracking, dominance-based crossover, and dominance-based gap-block shuffling. (See [7] and [8] for a description of these operators.) The reproduction process is repeated until enough children have been generated to replace all the parents. *Cofolga2mo* terminates when either a maximum number of iterations have been performed (1000 in our study), or when no new rank one solutions have been generated for a specified number of iterations (500 in our study).

Fig. 1. Relative improvement in terms of MS of the best solution on Pareto optimal frontiers over the best corresponding single objective when using four objectives.



6 Results

We have organized our experiments into five categories. First, we examine the performance of single objectives as compared with multiple objectives. Next, because the vast majority of the computational cost associated with generating multi-objective solutions is consumed

TABLE 5
Various performance metrics comparing Pareto frontiers generated by the Centroid, Cell and Sample coarsening methods with the exact frontiers.

Coarsening method	Coarsening parameter	Alignment time	Percent false positives	Frontier coverage	Best	Mean	BPO
Two objectives							
None	NA	33.416	0.000	100.000	0.732	0.666	0.696
Centroid	0.2	6.872	33.040	52.370	0.722	0.664	0.696
Centroid	0.05	6.367	33.997	54.163	0.728	0.667	0.696
Centroid	0.006	8.317	33.088	61.620	0.731	0.664	0.696
Cell	5	6.437	34.456	55.010	0.730	0.668	0.696
Cell	15	6.645	32.882	59.550	0.731	0.667	0.696
Cell	30	7.221	30.442	63.517	0.731	0.667	0.696
Sample	0.25	6.269	34.480	57.693	0.730	0.668	0.696
Sample	0.75	7.545	28.783	65.788	0.731	0.668	0.696
Sample	0.9	8.281	26.161	68.641	0.731	0.668	0.696
Three objectives							
None	NA	180.630	0.000	100.000	0.832	0.768	0.790
Centroid	0.2	4.992	58.585	31.307	0.809	0.756	0.790
Centroid	0.05	5.313	58.948	33.094	0.822	0.762	0.790
Centroid	0.006	66.892	50.781	45.522	0.829	0.762	0.790
Cell	5	4.887	57.797	35.764	0.826	0.766	0.790
Cell	15	6.332	52.534	42.729	0.829	0.766	0.790
Cell	30	8.989	47.216	48.278	0.829	0.766	0.790
Sample	0.25	4.667	58.268	35.841	0.825	0.768	0.790
Sample	0.75	6.348	51.408	43.934	0.828	0.769	0.790
Sample	0.9	7.944	46.789	48.243	0.829	0.769	0.790
Four objectives							
None	NA	241.771	0.000	100.000	0.868	0.805	0.822
Centroid	0.2	3.601	61.407	31.262	0.846	0.793	0.822
Centroid	0.05	7.675	61.828	32.691	0.858	0.798	0.822
Centroid	0.006	199.201	44.169	52.980	0.866	0.801	0.822
Cell	5	3.903	58.482	37.197	0.862	0.803	0.822
Cell	15	6.920	50.028	46.445	0.865	0.804	0.822
Cell	30	18.150	39.722	56.609	0.866	0.804	0.822
Sample	0.25	3.420	60.454	35.046	0.860	0.806	0.822
Sample	0.75	4.923	52.885	43.500	0.864	0.808	0.822
Sample	0.9	6.944	47.181	48.837	0.865	0.806	0.822

Description of columns: Best: the MS of the best solution on the Pareto frontier Mean: the mean MS of solutions on each Pareto frontier individually. BPO: Best performing objective. Rows with a coarsening method of "None" show results for the exact frontier.

in removing dominated solutions from consideration (the PF operation), we compare several algorithms to accomplish this. Next, since using multiple objectives can lead to multiple solutions, we compare methods that select a single solution from the Pareto frontier to be used as the result of the alignment. Then, we evaluate the Cell, Sample and Centroid methods, which reduce the number of solutions input to the PF operation. We conclude the presentation of our results with comparisons between the dynamic programming approach used here and the other alignment methods on both the ce_ref and BAliBASE RV11 datasets.

6.1 Single Vs Pareto Optimal Multiple Objectives

We generated Pareto optimal pairwise sequence alignments for all combinations of two, three and four objectives, and evaluated their performance on 588 protein pairs. Due to the large number of these experiments, in our results we focus our discussion on the five best performing combinations of two, three and four objectives. These results are shown in Table 4. Results for the top ten objectives can be found in Tables 3, 4 and 5 of the supplementary material. Note that due to the high computational requirements of generating exact frontiers for some protein pairs and objective combinations, we report results using the Cell reduction technique with its coarsening parameter set to ten. The relative performance of the other reduction techniques is explored in Section 6.3.

Comparing the best alignments generated by the multi-objective approaches with the alignments of the best corresponding single objectives (Table 4) we see that using multiple objectives in a Pareto framework can create better alignments. The match score achieved by the best two, three and four objective combinations is 0.761, 0.770 and 0.772, respectively. This represents an improvement of 4.5% to 6.0% over the corresponding best performing single objective. Moreover these improvements are statistically significant at $p < 0.1$ for two objectives, $p < 0.01$ for three objectives, and $p < 0.01$ for four objectives.

Examining the percent improved (FIM) column of Table 4, we see that for all combinations of objectives, the best alignment on the frontier has a higher match score than the top performing single objective in the combination for over 83% of protein pairs. To better illustrate this,

Figure 1 shows the performance of combinations of four objectives from Table 4 relative to their corresponding objectives. The plots in Figure 1 are histograms in which the height of each bar indicates the number of alignment pairs for which the best solution on the Pareto frontier has a MS that is better/worse than the MS of the best performing corresponding objective. Note that the pair of MS numbers were compared by taking their log-ratios. For all different combinations of objectives, the tallest bars appear to the right of zero and the area on the positive side is greatest, indicating that the best alignment on the Pareto optimal frontier frequently outperforms the corresponding single objective.

One interesting aspect of Figure 1 is the number of cases in which the best solution on the frontier does not outperform the best constituent objective (represented by bars to the left of the red line). Had our method been used to generate the exact frontier, these cases would not have arisen, since the best values for each objective individually are on that frontier. However, we use the Cell coarsening technique in these experiments, resulting in the occasional elimination of one or more of these solutions. In these cases, the frontier can lack a better solution than using the constituent objectives individually. To verify this, we looked at the 6409 exact frontiers generated for this experiment, and only 5 of them had a single best objective outperform the best solution on the exact frontier. In none of those cases was the difference more than 1%, and we attribute these to rare situations where very similar alignments have identical objective scores, but slightly different match scores.

6.2 Dominated Solution Elimination

To evaluate dominated solution elimination schemes (Section 4.1), we use the top performing objective combinations with a sample of protein pairs from the ce_ref dataset. Specifically, we use the top five performing objective combinations of size two, three, and four (the same combinations as those in Table 4), giving 15 total objective combinations. Using these combinations we align 98 randomly sampled protein pairs (one sixth of the full ce_ref set), with the different methods for eliminating dominated solutions. The amount of time required by the different methods is shown in Table 3.

Comparing the performance of the different schemes, we see that the Sort-all method requires 50% more time than the Merge-front method. This is not surprising, as the Merge-front method is equivalent to the Sort-all method, except that the Merge-front method takes advantage of the fact that a multi-objective dynamic programming alignment merges internally optimal Pareto frontiers. The Dominance-tree method is also slower than the Merge-front method, and even slower than the Sort-all method, as it assigns a dominance ranking to the input solutions, rather than simply identifying the Pareto frontier. The Merge-front-B method and the Merge-front-A method show only marginal performance gains over the Merge-front method. Whenever we report timings or speed-up factors for our methods, we refer to alignments produced using the Merge-front-B method.

6.3 Coarsening Optimizations

To study the impact the coarsening methods described in Section 4.2 on frontier generation, we use protein pairs that can have their exact frontiers generated in a reasonable amount of time (less than 30 minutes), but not so quickly that coarsening is not necessary. This constrains the required computational time, while still providing meaningful data for examining coarsening methods. For combinations of objectives, we use the top ten objectives for sets of two, three and four, for a total of 30 objective sets. All told we compare 2671, 2222 and 1516 alignments for objective combinations of size two, three and four, respectively. We examine a range of parameters for each of the Cell, Centroid and Sample coarsening techniques, and list a selection of the results in Table 5, with the full results of the study available in Table 2 of the supplementary material.

In terms of the time required to produce a solution with a given quality, the Sample technique is on average 2.98, 27.88, and 58.07 times faster than generating the entire Pareto optimal frontier when using sets of two, three and four objectives, respectively. The Cell technique is also much faster, specifically 2.88, 21.88 and 33.59 times faster.

In terms of the quality of the Pareto frontier, we see a graceful degradation of coverage and a gradual increase in false positives with an increased level of approximation through the use of the coarsening parameter. However, these metrics are not closely tied to the match scores found on the frontiers, and as such the average match scores over alignments on the frontiers change very little as a result of different coarsening parameters. With respect to the coarsening methods, both the Sample and Cell methods show smooth changes in run times according to the coarsening parameter, while the results for the Centroid method show more severe changes, particularly in the case of four objectives.

The relationship between speed increase and alignment quality is illustrated in Figures 2 and 3. Each point in Figure 2 represents a protein pair and is plotted based on how much more quickly the Sample method with its coarsening parameter set to 0.25 generates a Pareto optimal frontier. Figure 3 plots the match score of the best solution on the approximate frontier relative to that of the best solution on the exact frontier. The lengths of the vertical lines are determined as

$$\log\left(\frac{\text{MS of best on approximate frontier}}{\text{MS of best on exact frontier}}\right).$$

From the results in Figure 2 we see that the speedups obtained by the coarsening scheme varies widely for different alignment pairs, from less than an order of magnitude to nearly three orders of magnitude. Although the

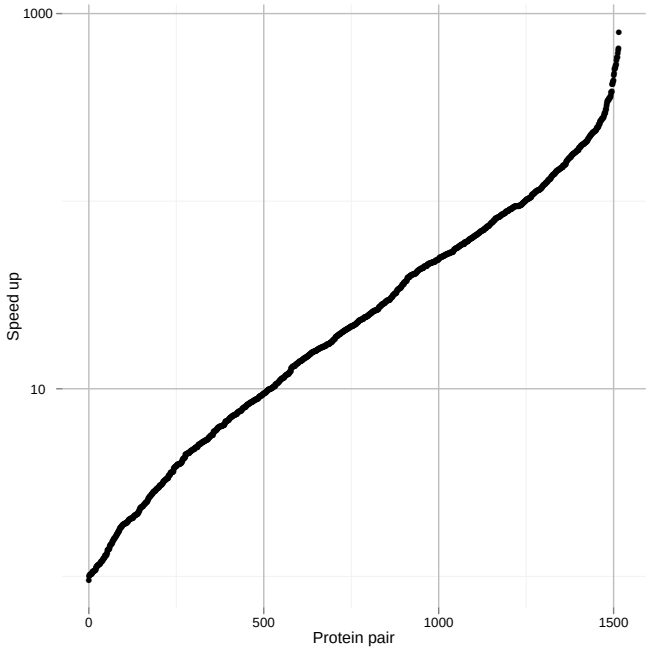


Fig. 2. Four objective coarsening algorithm performance. A speed up of X indicates that the Sample method with coarsening parameter 0.25 was X times faster than generating the exact frontier.

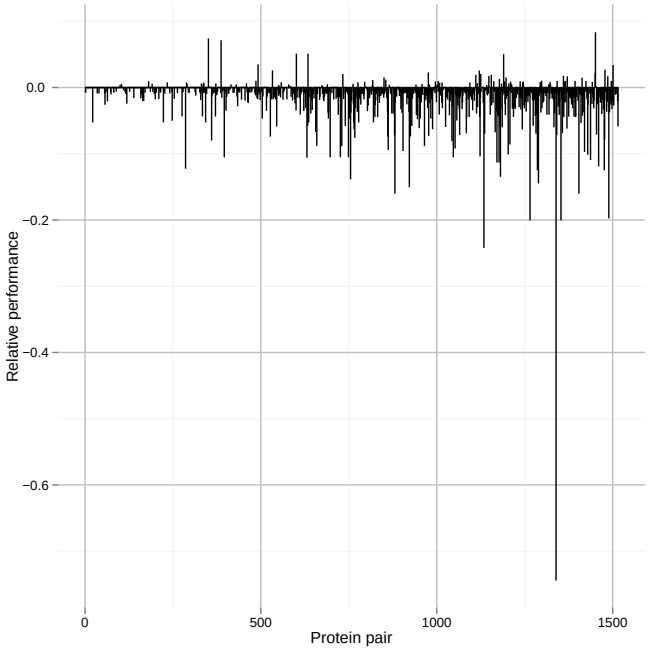


Fig. 3. Four objective coarsening algorithm performance. The lengths of the vertical lines are determined as $\log\left(\frac{\text{MS of best on approximate frontier}}{\text{MS of best on exact frontier}}\right)$.

TABLE 6
Various performance metrics comparing a selected solution on the Pareto frontier with the solution of the best corresponding single objective.

Objectives	Unit-space MS/FIM	Unit-z MS/FIM	OPW MS/FIM	Objectives	Unit-space MS/FIM	Unit-z MS/FIM	OPW MS/FIM
SS Pc	0.730/0.405	0.724/0.400	0.730/0.405	SS Pc Cp	0.729/0.381	0.726/0.372	0.728/0.384
SS Cf	0.723/0.393	0.728/0.395	0.725/0.386	SS Pc Rp	0.730/0.430	0.728/0.403	0.730/0.425
Pc Cp	0.729/0.369	0.722/0.350	0.729/0.371	SS Pc Cf	0.733/0.427	0.734/0.417	0.733/0.427
Pc Rp	0.728/0.362	0.725/0.395	0.728/0.364	SS Cp Rf	0.726/0.446	0.726/0.437	0.726/0.447
Cf Cp	0.722/0.362	0.723/0.354	0.722/0.359	SS Cf Cp	0.727/0.383	0.728/0.354	0.727/0.359
SS Pc Cp Rp	0.729/0.403	0.727/0.386	0.730/0.398				
SS Pc Cp Rf	0.728/0.372	0.726/0.364	0.727/0.367				
SS Pc Cf Cp	0.730/0.388	0.730/0.374	0.730/0.388				
SS Pc Cf Rp	0.732/0.401	0.731/0.391	0.732/0.413				
SS Cf Cp Rp	0.727/0.398	0.728/0.415	0.727/0.406				

Abbreviations for objectives: SS: SS_dotp Pc: Picasso Cf: Correlf Cp: Correlp Rf: Rankf Rp: Rankp
Description of columns: OPW: Objective Performance Weighted. MS: Match score. FIM: Fraction improved.
Values other than FIM are average match scores over 588 protein pairs

TABLE 7
Various performance metrics comparing a solution generated by a linear combination of objectives and the best solution generated by the evolutionary algorithm with the solutions of the best corresponding single objective.

Objectives	Linear MS/FIM	EA MS/FIM	Objectives	Linear MS/FIM	EA MS/FIM	Objectives	Linear MS/FIM	EA MS/FIM
SS Pc	0.717/0.250	0.732/0.070	SS Pc Cp	0.727/0.372	0.742/0.297	SS Pc Cp Rp	0.724/0.366	0.749/0.484
SS Cf	0.719/0.163	0.729/0.055	SS Pc Rp	0.724/0.366	0.744/0.416	SS Pc Cp Rf	0.727/0.381	0.748/0.527
Pc Cp	0.724/0.313	0.740/0.256	SS Pc Cf	0.722/0.366	0.742/0.406	SS Pc Cf Cp	0.724/0.378	0.748/0.504
Pc Rp	0.724/0.253	0.741/0.389	SS Cp Rf	0.713/0.417	0.729/0.338	SS Pc Cf Rp	0.722/0.366	0.749/0.566
Cf Cp	0.723/0.337	0.735/0.264	SS Cf Cp	0.718/0.284	0.738/0.307	SS Cf Cp Rp	0.720/0.367	0.745/0.506

Abbreviations for objectives: SS: SS_dotp Pc: Picasso Cf: Correlf Cp: Correlp Rf: Rankf Rp: Rankp
Description of columns: Linear: A linear combination of objectives. EA: Evolutionary algorithm. MS: Match score. FIM: Fraction improved.
Values other than FIM are average match scores over 588 protein pairs

TABLE 8
Various performance metrics describing the runtime and quality of the solutions generated by the evolutionary algorithm.

Number of objectives	Alignment time	Percent false positives	Frontier coverage	Unit-space	OPW	Best	Mean
2	1282.717	96.731	3.154	0.678	0.694	0.703	0.643
3	1882.757	96.671	3.168	0.790	0.790	0.804	0.748
4	1820.837	95.382	4.117	0.825	0.826	0.842	0.795

Description of columns: Linear: a linear combination of objectives. EA: Evolutionary algorithm. FIM: Fraction improved. OPW: Objective Performance Weighted. Best: The MS of the best solution generated by the evolutionary algorithm. Mean: The average MS generated by the evolutionary algorithm. We have omitted the Unit-z method results due to space constraints and their close similarity to the Unit-space method

overlap between the approximate and exact frontiers is small (see Table 5), the overall degradation in the quality of the alignments (as measured by their match score) is also small. However, there is an increase in the relative quality degradation with increased speedup. This is to be expected, as in these cases there is a higher degree of approximation. Note that it is possible for the alignment on the approximate frontier to be better with respect to match score because this is not the value being optimized by the alignment process.

6.4 Solution Selection

The results in Table 4 indicate that high-quality solutions to the global sequence alignment problem exist on the Pareto frontier, but they also show that for a given frontier, the mean match score is usually lower than the best match score. Table 6 shows the performance of the various solution selection schemes described in Section 4.3 over the 588 protein pairs from the ce_ref set. Comparing Table 6 with Table 4, we see that all three of the selection schemes (Unit-space, Unit-z and Objective Performance Weighted) do better than the mean match score. However, their performance is worse than the performance achieved by the best corresponding single objective.

In quite a few cases, the average match score is slightly worse than that of the best corresponding single objective (though the best on the frontier tends to be superior to it). These results suggest that although the Pareto optimal frontier contains some high quality solutions, being able to select the one that achieves the best match score is non-trivial.

6.5 Comparisons with other Algorithms

Table 7 shows the performance of linear combinations of objectives and the best solution generated by the evolutionary algorithm (see Section 5.6.2). Note that these results are not directly comparable with one another, as the linear combination results in a single alignment, while the evolutionary algorithm produces many alignments, among which the one that achieves the best match score is selected and reported.

Comparing the results for the linear combination of objectives (Table 7) with the selection methods (Table 6), we see that for both the average match score and FIM values the selection methods show slightly better match scores than the linear combination, irrespective of the number of objectives involved. Additionally, comparing Tables 2 and 7 we see that linear combinations of objectives produce worse results than Picasso in isolation, with a selected alignment from the Pareto optimal frontier being marginally better.

Comparing the results for the evolutionary algorithm (Table 7) with the "Best" column in Table 4, we see that for both the average match score and FIM values the best solution on the Pareto optimal frontier is superior to the best alignment generated by the evolutionary algorithm. This is the case for all combinations of objectives, regardless of the number of objectives involved. These gains are considerably bigger than those of the selection methods over the linear combinations of objectives. However, it is important to note that at this point the best alignment on the frontier cannot be reliably selected from the rest of the frontier without knowing the true alignment (see Section 6.4).

Regarding the evolutionary algorithm as a method for generating a Pareto frontier of alignments, few of the alignments generated by this approach are on the exact Pareto frontier, regardless of the combinations of objectives used and despite the long running times. With four objectives, the evolutionary approach produces around 4% coverage in around 1820 seconds (see Table 8) which is considerably more than the time required for the dynamic programming approach to generate the exact Pareto optimal frontier (around 241 seconds). The evolutionary approach only rarely generates alignments with higher objective scores than those obtained from single-objective measures, while our Sample approach frequently generates them. The Sample technique generates 57.69%, 35.84% and 35.05% coverage of the frontier for sets of two, three and four objectives (Table 5), while the evolutionary algorithm generates 3.15%, 3.17% and 4.12% (Table 8). An interesting observation from the FIM values in Tables 4, 6 and 7 is that neither the evolutionary algorithm, a linear combination of objectives, nor selecting an alignment from a Pareto optimal frontier consistently outperforms the best corresponding single objective.

6.5.1 Performance Summary

Table 9 shows the performance of the best objective combination on ce_ref for each alignment method, and the performance of that alignment method and objective

combination on the RV11 dataset. Examining this table, we see that there is little difference in performance between the various alignment methods. The one exception is the best solution on the Pareto frontier, but as previously mentioned this solution cannot yet be reliably identified without knowing the true alignment. This is true for both the ce_ref dataset and the RV11 dataset from BAliBASE. Comparing the performance between these two datasets, we see that the performance on RV11 is considerably worse than that on the ce_ref set. However, the relative performance of the various techniques within a dataset is consistent–all of the techniques are roughly on par with one another. The performance is also in-line with a recent study on this dataset [34] in which ClustalW achieves an SPS score of 58.16. (The SPS score is identical to the match score except that it only considers core regions of an alignment).

TABLE 9
Best performing objective combinations for pairwise alignment methods on ce_ref and BAliBASE RV11

Alignment method	Score (ce_ref)	Score (BAliBASE RV11)
Picasso alone	0.728	0.569
Linear combination	0.727	0.562
Evolutionary (Unit-z)	0.725	0.566
Evolutionary (best)	0.751	0.587
Pareto (Unit-z)	0.733	0.568
Pareto (best)	0.772	0.615

Values in the second column represent the average over alignments between 588 protein pairs; values in the third column represent the average over alignments between 913 different protein pairs. Objectives combinations for each method: Pareto (best): SS_dotp Picasso Correlp Rankp pareto (Unit-z): SS_dotp Picasso Correlf Evolutionary (best): Picasso Correlf Correlp Rankp Evolutionary (Unit-z): SS_dotp Picasso Correlf Linear: SS_dotp Picasso Correlp Rankf

7 Discussion and Conclusion

We have presented a technique to align pairs of protein sequences using multiple profile-profile scoring functions in a dynamic programming framework to produce Pareto optimal frontiers of alignments, and techniques to select a good alignment from such a frontier. Within this framework, we have shown how to reduce the time spent eliminating dominated solutions, and heuristic techniques to reduce the sizes of intermediate frontiers.

The results presented here indicate that multi-objective optimization using Pareto optimality leads to Pareto frontiers that contain higher quality alignments than those produced by other multi-objective approaches and by single objectives. However, selecting a high-quality solution is challenging, and the methods we have presented do not consistently select the best alignment. In regard to the selection schemes, both the Unit-space and Unit-z selection schemes perform on par with the Objective Performance Weighted scheme, even though the results for the Objective Performance Weighted scheme relies upon knowing the relative performance of the scoring functions on the entire data set. Thus, it is encouraging that these schemes show comparable performance without such a requirement.

Concerning the elimination of dominated solutions, implementing the Merge-front-B method results in modest performance improvements, as it is not often that only one of $\mathcal{E}$ or $\mathcal{F}$ is used in the construction of $\mathcal{V}$ (see Section 4), a condition which is required for Merge-front-B to have an advantage. In contrast, the condition that Merge-front-A attempts to leverage is somewhat common, but ironically difficult to detect. As shown in Table 3, the cost associated with detecting this condition actually overshadows the potential gains in performance.

When aligning protein sequences using the SS_dotp objective in conjunction with other objective functions, an interesting trend appears. Despite its poor performance when used as a single objective, the SS_dotp objective is part of several top-performing combinations of objectives. This is the case for both linear and Pareto multi-objective formulations. On the one hand, the vector of predictions for an amino acid contains only three values, and this could explain why using SS_dotp in isolation does not perform well. On the other hand, secondary structure predictions are made by considering a window around an amino acid, which includes more information about its local environment, and may help to explain the boost in performance it provides.

Due to the considerable diversity in the quality of the solutions on the Pareto frontier, improving techniques for selecting a high-quality solution is a direction for future research. Another direction involves leveraging the information contained within the Pareto frontiers. The number and nature of solutions found on a Pareto frontier contain considerably more information than a single-objective alignment, and can potentially help explain the relationship between the input sequences. For example, multiple frontiers of alignments from different sequences could serve as input to a multiple sequence alignment program, resulting in better alignments. Also, agreement between profile-profile scoring functions results in fewer alignments on the Pareto frontier than when such functions disagree, so the number of alignments produced indicates the strength of the evolutionary signals in the profiles. This could be used to produce more sensitive algorithms for functional site identification by looking at regions with particularly strong signals, for remote homology detection, or for building better profiles.

Acknowledgments

This work was supported in part by access to research and computing facilities provided by the Digital Technology Center and the Minnesota Supercomputing Institute.

References

[1] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman, "Gapped blast and psi-blast: A new generation of protein database search programs," *Nucl. Acids Res.*, vol. 25, no. 17, pp. 3389–3402, 1997. [Online]. Available: http://nar.oxfordjournals.org/cgi/content/abstract/25/17/3389

[2] R. Sadreyev and N. Grishin, "Compass: a tool for comparison of multiple protein alignments with assessment of statistical significance." *Journal of Molecular Biology*, vol. 326, no. 1, pp. 317–336, 2003. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0022283602013712

[3] R. C. Edgar and K. Sjölander, "A comparison of scoring functions for protein sequence profile alignment," *Bioinformatics*, vol. 20, pp. 1301–1308, 2004.

[4] J. D. Thompson, F. Plewniak, and O. Poch, "Balibase: a benchmark alignment database for the evaluation of multiple alignment programs." *Bioinformatics*, vol. 15, no. 1, pp. 87–88, 1999.

[5] O. Gotoh, "An improved algorithm for matching biological sequences," *Journal of Molecular Biology*, vol. 162, pp. 705–708, 1982.

[6] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. New York, NY: Cambridge University Press, 1997.

[7] A. Taneda, "Multi-objective pairwise rna sequence alignment." *Bioinformatics*, vol. 26, no. 19, pp. 2383–2390, Oct 2010.

[8] ——, "An efficient genetic algorithm for structural rna pairwise alignment and its application to non-coding rna discovery in yeast." *BMC Bioinformatics*, vol. 9, p. 521, 2008.

[9] P. Seeluangsawat and P. Chongstitvatana, "A multiple objective evolutionary algorithm for multiple sequence alignment," *Proceedings of the 2005 conference on Genetic and evolutionary computation GECCO 05*, vol. 1, p. 477, 2005.

[10] F. J. M. da Silva, J. M. S. Pérez, J. A. G. Pulido, and M. A. V. Rodríguez, "Parallel niche pareto alineaga–an evolutionary multiobjective approach on multiple sequence alignment." *J Integr Bioinform*, vol. 8, no. 3, p. 174, 2011.

[11] M. A. Roytberg, M. N. Semionenkov, and O. I. Tabolina, "Pareto-optimal alignment of biological sequences," *Biofizika*, vol. 44, no. 4, pp. 581–594, 1999.

[12] D. Gusfield, K. Balasubramanian, and D. Naor, "Parametric optimization of sequence alignment," in *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '92. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1992, pp. 432–439.

[13] L. Paquete and J. P. O. Almeida, "Experiments with bicriteria sequence alignment," in *Cutting-Edge Research Topics on Multiple Criteria Decision Making*, ser. Communications in Computer and Information Science, Y. Shi, S. Wang, Y. Peng, J. Li, and Y. Zeng, Eds. Springer Berlin Heidelberg, 2009, vol. 35, pp. 45–51.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182 –197, apr 2002.

[15] C. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.

[16] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary Multiobjective Optimization*, ser. Advanced Information and Knowledge Processing, A. Abraham, L. Jain, R. Goldberg, L. C. Jain, and X. Wu, Eds. Springer Berlin Heidelberg, 2005, pp. 105–145.

[17] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: Amosa," *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 3, pp. 269 –283, june 2008.

[18] Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 6, pp. 658 –675, dec. 2006.

[19] A. H.-C. Mohamed Ben Othman and G. A. Azim, "Genetic algorithms and scalar product forpairwise sequence alignment," *International Journal of Computers*, vol. 2, no. 2, 2008.

[20] G. Garai and B. Chowdhury, "A novel genetic approach for optimized biological sequence alignment," *Journal of Biophysical Chemistry*, vol. 3, pp. 201–205, 2012.

[21] C. Notredame, E. A. O'Brien, and D. G. Higgins, "Raga: Rna sequence alignment by genetic algorithm," *Nucleic Acids Research*, vol. 25, no. 22, pp. 4570–4580, 1997.

[22] C. Zhang and A. Wong, "Toward efficient multiple molecular sequence alignment: a system of genetic algorithm and dynamic programming," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 27, no. 6, pp. 918 –932, dec 1997.

[23] A. L. Jaimes and C. A. C. Coello, "An introduction to multiobjective evolutionary algorithms and some of their potential uses in biology," in *Applications of Computational Intelligence in Biology*, ser. Studies in Computational Intelligence, T. G. Smolinski, M. G. Milanova, and A. E. Hassanien, Eds. Springer, 2008, vol. 122, pp. 79–102.

[24] M. T. Jensen, "Reducing the run-time complexity of multiobjective eas: The nsga-ii and other algorithms," *IEEE Trans. Evolutionary Computation*, vol. 7, no. 5, pp. 503–515, 2003.

[25] H. Fang, Q. Wang, Y.-C. Tu, and M. F. Horstemeyer, "An efficient non-dominated sorting method for evolutionary algorithms," *Evolutionary Computation*, vol. 16, no. 3, pp. 355–384, 2008.

[26] J. C. Ferreira, C. M. Fonseca, and A. Gaspar-Cunha, "Methodology to select solutions from the pareto-optimal set: a comparative study," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, ser. GECCO '07. New York, NY, USA: ACM, 2007, pp. 789–796.

[27] I. Shindyalov and P. E. Bourne, "Protein structure alignment by incremental combinatorial extension (ce) of the optimal path," *Protein Engineering*, vol. 11, pp. 739–747, 1998.

[28] A. S. Konagurthu, J. C. Whisstock, P. J. Stuckey, and A. M. Lesk, "Mustang: a multiple structural alignment algorithm." *Proteins*, vol. 64, no. 3, pp. 559–574, Aug 2006.

[29] C. Kemena and C. Notredame, "Upcoming challenges for multiple sequence alignment methods in the high-throughput era," *Bioinformatics*, vol. 25, no. 19, pp. 2455–2465, 2009.

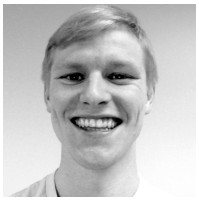
[30] H.-N. Lin, C. Notredame, J.-M. Chang, T.-Y. Sung, and W.-L. Hsu, "Improving the alignment quality of consistency based aligners with an evaluation function using synonymous protein words," *PLoS ONE*, vol. 6, no. 12, p. e27872, 12 2011.

[31] A. Heger and L. Holm, "Picasso: generating a covering set of protein family profiles," *Bioinformatics*, vol. 17, no. 3, pp. 272–279, 2001.

[32] G. Karypis, "Yasspp: Better kernels and coding schemes lead to improvements in svm-based secondary structure prediction," *Proteins: Structure, Function and Bioinformatics*, vol. 64, no. 3, pp. 575–586, 2006.

[33] R. H. M. Cline and K. Karplus, "Predicting reliable regions in protein sequence alignments," *Bioinformatics*, vol. 18, pp. 306–314, 2002.

[34] Y. Liu, B. Schmidt, and D. L. Maskell, "Msaprobs: multiple sequence alignment based on pair hidden markov models and partition function posterior prob abilities," *Bioinformatics*, vol. 26, no. 16, pp. 1958–1964, 2010.



Kevin W. DeRonne Kevin W. DeRonne is a doctoral student at the Department of Computer Science & Engineering at the University of Minnesota, Twin Cities. His research interests include structural bioinformatics, information retrieval and computer vision.

George Karypis George Karypis is a professor at the Department of Computer Science & Engineering at the University of Minnesota, Twin Cities. His research interests span the areas of data mining, bioinformatics, cheminformatics, high performance computing, information retrieval, collaborative filtering, and scientific computing. He has coauthored over 210 papers on these topics and two books ("Introduction to Protein Structure Prediction: Methods and Algorithms" (Wiley, 2010) and "Introduction to Parallel Computing" (Publ. Addison Wesley, 2003, 2nd edition)). In addition, he is serving on the editorial board of many journals and program committees of many conferences and workshops on these topics.