Optimization and Machine Learning Applications to Protein Sequence and Structure

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Kevin W. DeRonne

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

George Karypis

January, 2013

Acknowledgements

This thesis represents the culmination of over ten years of work. Not once over those ten years did I actually expect to be writing these words, and it is only through the stalwart, foolish, patient love and support of a great many people that I am. There are far too many names I need to list, so I can only name a few, but to the others who do not appear here: you are by no means forgotten.

Professors Cavan Reilly, Ravi Janardan and Vipin Kumar: thank you for taking the time to serve on my preliminary and final examination committees. It is an honor to have people of such quality reviewing my work.

Professor George Karypis: I have no idea why you put up with me for so long, but here we are. Despite my repeated attempts to force you to give up on me, you never did. The only way I can be succinct about what you have done for me is this: I have become the computer scientist I am because of you.

Patrick Coleman Saunders: in death as in life, you made me a better person.

Chas Kennedy, David Mulvihill, Dan O'Brien, Adam Meyer, Andrew Owen, Huzefa Rangwala, Jeff Rostis, David Seelig, Neil Shah and Nikil Wale: I hope you know that you are all like brothers to me. You have all done your part to keep me (mostly) sane for the last decade. I also need to thank mi hermano Christopher Kauffman for all his love, help and support.

Momb and Pop: You never forced me to do anything, you supported me in all my decisions, and you made this work possible. I owe you more than I can ever repay, but this work is a (very, very) small attempt at that. Jane and Paul: Thanks for your unconditional love, for being on "the team", for motivating me to get things done, and for understanding when I didn't.

Dedication

To Beth, for telling me I could quit when I needed to hear it, even though you would never let that happen. For allowing me to be manic and angry and unreasonable, even at the worst possible times. For loving and caring for me, even when I couldn't do either myself. For all of this, even though I didn't deserve any of it: Thank you.

Abstract

Algorithms that enable the development of drugs to inhibit or enhance protein functions save time, money and effort spent on bench research. This dissertation presents algorithms for protein structure prediction, and for the prediction of residues that form protein-protein interactions. Within the context of protein structure prediction, we present algorithms for sequence alignment, for the optimization of fragments into complete structures, and for the assessment of predicted structure quality.

We demonstrate the utility of incorporating multiple objectives when aligning pairs of protein sequence profiles. We present a proof that the problem of generating Pareto optimal pairwise alignments has the optimal substructure property, and we present an efficient algorithm for generating Pareto optimal frontiers of pairwise alignments. Despite the efficiency of our exact algorithm, for certain pairs of sequences the computational cost remains high. To address this, we developed a heuristic approach to produce approximated Pareto optimal frontiers of pairwise alignments. The frontiers our algorithm produces 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 that are 6% better than the alignments obtained by single objectives. We have provided a theoretically sound way of combining multiple objectives when aligning pairs of sequences.

Assembling fragments of known structures to form complete proteins is a key technique for predicting the structures of novel protein folds. Several existing methods use stochastic optimization methods to assemble fragments. We examine deterministic algorithms for optimizing scoring functions in protein structure prediction. We present a technique that can overcome local minima, and determine the extent to which these minima affect the optimization. Experiments on a diverse set of proteins show that

our algorithms consistently outperform existing approaches, producing results 6-20% better. Our work in fragment assembly optimization has enabled the development of better protein structure prediction algorithms.

We also present methods that can automatically assess the quality of computationally predicted protein structures. We examine techniques to estimate the quality of a predicted protein structure based on prediction consensus. The structure being assessed is aligned to different predictions for the same protein using local-global alignment. On two datasets, we examine both static and machine learning methods for aggregating distances between residues within these alignments. We find that a constrained regression approach shows performance improvements of over 20%, and can be easily retrained to accommodate changing predictors. Our algorithm for model quality assessment enables the effective use of multiple structure prediction techniques.

With respect to predicting interacting residues, we present a method that uses high quality sequence alignments to identify protein residues that bind to other proteins. In contrast to existing approaches, which focus on local sequence information, our method uses global sequence information from induced multiple sequence alignments to make better predictions. On a large and challenging dataset, our method achieves an area under the receiver-operating characteristic (ROC) of 0.656, compared to 0.615 achieved by the existing ISIS technique. By leveraging *a priori* measures of alignment quality, we can further increase performance to 0.768 ROC on a subset of the data. Our algorithms have allowed for better manipulation of protein function.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Living organisms use proteins to perform a wide array of functions, from metabolism to cellular reproduction to the construction of proteins themselves. The specific function for a given protein depends greatly on its structure; thus, better structural information leads to a better understanding of intricate biological processes.

Proteins consist of sequences of amino acids, also called *residues*, which fold into structures. With recent advances in large scale sequencing technologies, we have seen an exponential growth in protein sequence information, but it is the structure of a protein that dictates its function. Currently, our ability to identify the sequences of proteins far out-paces the rate at which we can identify their three-dimensional structures. Understanding the relationship between sequence and three dimensional structure remains a fundamental problem in molecular biology [1]. Biologists increasingly rely on computational techniques to extract useful information from known structures contained in large databases, and the techniques themselves remain an active area of research.

Proteins often perform their biological function by binding to other proteins in order to form various protein complexes. Knowing the protein residues involved in these protein-protein interactions is essential to both understanding protein functions and developing rational approaches for manipulating such interactions.

1.1 Objectives and their Significance

There are two primary objectives of this dissertation. The first objective is to advance the field of protein structure prediction. The second objective is to advance the field of interacting residue prediction. We approach the first objective from three directions. To begin, we develop better algorithms for aligning protein sequences. Next, we develop better algorithms for optimizing the assembly of protein fragments into a protein structure prediction. Finally, we develop better algorithms to assess the quality of a structure prediction. We approach the second objective directly, developing a better algorithm to predict, solely from sequence, which of a protein's residues interact with other proteins.

Knowing the structure of a protein enables the inference of, among other things, its function, its cellular location, its stability and its vulnerabilities. All of these can be important when developing new agents to treat and cure disease, when developing new crop strains and when developing new catalysts, to name just a few of the applications. The ability to predict a protein structure purely from its sequence is so important that it has been called the "holy grail" of bioinformatics [2, 3, 4, 5].

Many proteins can only function through interacting with other proteins, so knowing which residues are involved in protein-protein interactions is essential when attempting to enhance or inhibit protein functions. For example, it can be desirable to inhibit the function of a protein that decreases the effectiveness of a certain drug, or enhance the function of a protein that increases the effectiveness of that drug.

1.2 Contributions

1.2.1 Protein Structure Prediction

The biennial Critical Assessment of Structure Prediction (CASP) competition[1] breaks down structure prediction into three categories, listed here in increasing order of difficulty: homologous[2] fold recognition, analogous fold prediction and new fold prediction. In homologous fold recognition the structure of the unknown, or *query* sequence, is

[1] For a complete explanation of this competition, see `http://predictioncenter.org`

[2] Homologs are proteins related through a common evolutionary ancestor.

similar to a *template* protein with known structure. However, the template and query sequences have a low, though detectable, similarity. In analogous fold recognition there exists a known structure similar to the correct structure of the query, but the template sequence has no detectable similarity to the query sequence. Still more challenging is the problem of new fold prediction: predicting the structure of a query sequence lacking a homolog with a known structure.

There are three general classes of methods for protein structure prediction, corresponding to the three problems outlined by the CASP competition. Homologous fold recognition needs only comparative modeling techniques to be solved. Analogous fold prediction can be solved using threading techniques, where a sequence is "threaded" into a template structure. New fold prediction needs more advanced methods that build the structure from scratch, such as fragment assembly and techniques in computational biology.

Pareto optimal multi-objective alignments In comparative modeling, the basic idea is that a pair of sequences with sufficient similarity can be assumed to have the same structure. In this technique, the first step is to identify suitable template proteins for the query. After one or more templates have been selected, sequence alignment methods can be used to map the residues of the query sequence to the residues of a given template. Using this mapping, coordinates from the known structure are refined within the context of the query sequence to account for differences between the query and the template.

Low-quality alignments between the query and template sequences can misidentify a sequence as being homologous to the query, or miss homologs altogether. Consequently, algorithms that produce high quality alignments are a critical component of protein structure research. Previous work in protein sequence alignment uses a single source of evolutionary information for determining the quality of aligning a pair of amino acids.

The first contribution of this dissertation is the development of methods that combine multiple objectives for sequence alignments. This is done using a Pareto optimal framework to combine multiple sources of evolutionary information for protein sequence alignments. Within the context of this work, we developed methods to addresses computational challenges surrounding the production of these alignments.

We begin our work on sequence alignment by proving that the problem of generating multi-objective Pareto optimal sequence alignments has the optimal substructure property. Relying on this proof, we present a dynamic programming sequence alignment algorithm that can simultaneously use multiple metrics for determining the quality of aligning a pair of amino acids, and which generates a Pareto optimal frontier of alignments. We present a highly efficient algorithm for eliminating suboptimal alignments, which is the most computationally intensive part of the alignment algorithm. Our elimination technique is 60% faster than any previously published algorithm. However, the time required to produce a Pareto optimal frontier of alignments remains quite long. We further reduce the amount of time spent generating alignments through a heuristic method that reduces the number of alignments which are input to the elimination algorithm. The heuristic steps produce alignments of equal quality while running two orders of magnitude faster.

Our algorithms for Pareto optimal sequence alignments create alignments that are 6% better than alignments using single objectives alone, or linear combinations of alignments. In addition, our algorithms generate exact Pareto optimal frontiers in less than a quarter of the time required by existing evolutionary algorithms to generate an inexact frontier. Our efforts have provided a theoretically sound way of combining multiple objectives when aligning pairs of sequences.

Fragment assembly optimization Methods for new fold prediction generally rely on assembling fragments of known structures into a complete protein, using a *scoring function* to assess the quality of placing a fragment at a given position in the structure. Such functions typically estimate the free energy in the structure, as true protein structures have minimal free energy [6]. With a high quality scoring function, the problem becomes that of optimizing the function's value, the *score* of the structure, through the placement of structural fragments. However, given a protein with a reasonable number of amino acids (a_n) and a reasonable number of fragments corresponding to each amino acid (f_n), the number of possible structural configurations is very large, being proportional to ${a_n}^{(f_n)}$. Additionally, there may be many local minima in the range of the scoring function, further complicating the task of assembling fragments. Producing high quality structures, then, requires an efficient optimization of the scoring function

through fragment assembly.

The second contribution of this dissertation is the development of methods that optimize a fragment assembly scoring function more efficiently than previous techniques. A key aspect of our algorithm is its ability to overcome local minima, which results in objective function values up to 20% better than those obtained using the previous leading technique. When assembling protein fragments, detecting atomic conflicts is the most time-consuming step. We present an efficient method for detecting such conflicts which computes the minimum number of atomic distances to ensure a structure without conflicts. Our work in fragment assembly optimization has enabled the development of better protein structure prediction algorithms.

Model quality assessment Rather than a single prediction, many structure prediction algorithms produce an ensemble of structures. Even when considering all structures in such an ensemble, the true structure for a query sequence is rarely, if ever, predicted correctly. This is due to the large number of possible fragments, an even larger number of fragment configurations, and imperfect scoring functions. Ultimately, only a single prediction can be used as the model for a query. Key to the success of any protein structure prediction framework, then, are methods to assess the quality of a predicted structure. Such methods typically rely on the intrinsic properties of the prediction, or on comparing the properties of multiple predictions to assess their quality.

The third contribution of this dissertation is the development of techniques to estimate the quality of a predicted protein structure based on the consensus between multiple structure prediction algorithms. Local-global alignment (LGA) is used to align a query structure to various predicted structures for the same protein. We use both static and machine learning methods for aggregating the results from LGA to produce an estimate of structure quality. We find that a constrained regression approach outperforms other machine learning methods, and static averaging methods by over 20%. This is the first time such approaches have been used to estimate protein model quality.[3] Our algorithm for model quality assessment enables the effective use of multiple structure prediction techniques, and techniques that produce multiple structures for a query.

[3] The original publication for this work earned the "highly accessed" label on BMC Bioinformatics.

1.2.2 Interacting Residue Prediction

Many techniques have been developed to predict protein-protein interactions. Some of these techniques rely only on protein sequence, and some rely on protein structure. Those that rely on structure can form more accurate predictions from the additional information, but those that only require sequence are more useful, as there are more known sequences than known structures. Existing methods incorporate information local to each residue, such as the physiochemical properties of the residue and its immediate environment, while more global information from sequence alignments has not yet been effectively utilized.

The fourth contribution of this dissertation is the development of a new method for using high quality sequence alignments to predict which amino acids in a sequence interact with other proteins. Our method for interacting residue prediction uses global sequence information from induced multiple sequence alignments to make predictions. This improves upon existing approaches which focus on local sequence information. Our method achieves an area under the receiver-operating characteristic curve (ROC) of 0.656, compared to 0.615 achieved by ISIS, an existing technique. By leveraging *a priori* measures of alignment quality, we can further increase performance to 0.768 ROC on a subset of the data. Our algorithms have allowed for better manipulation of protein function.

1.3 Related Publications

The work presented in this thesis has been published in leading journals an at conferences in bioinformatics. The related publications are listed below.

- Chapter 2: **Kevin W. DeRonne** and George Karypis. Pareto Optimal Pairwise Sequence Alignment. In *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2013
- Chapter 4: **Kevin W. DeRonne** and George Karypis. Improved Estimation of Structure Predictor Quality. In *BMC Structural Biology*, 2009
- **Kevin W. DeRonne**, Huzefa Rangwala, and George Karypis. Protein Structure Prediction using String Kernels. In *Knowledge Discovery in Bioinformatics*, 2007

- Chapter 3: **Kevin W. DeRonne** and George Karypis. Effective Optimization Algorithms for Fragment-Assembly Based Protein Structure Prediction. In *Journal of Bioinformatics and Computational Biology*, 2007.

- **Kevin W. DeRonne** and George Karypis. Effective Optimization Algorithms for Fragment-Assembly Based Protein Structure Prediction. In *Computational Systems Bioinformatics Conference*, 2006.

Chapter 2

Pareto Optimal Multi-objective Alignment

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 scoring schemes have been developed whose individual performance has been extensively compared [7, 8, 9]. In this chapter we demonstrate that better 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.

We present 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 [10].

This is currently 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.

2.1 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 [11, 12], multiple sequence alignment [13, 14] and treating gap penalties as an objective to be optimized [15, 16, 17]. 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 [18, 19, 20, 21, 22], and also to perform sequence alignments using a single objective [23, 24, 25, 26] or multiple objectives [27, 11, 12, 13, 14]. 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 either the new solutions, the previous solutions, or a combination of the two. This process is repeated until some pre-determined 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 [14] and using dominance-rank assignment [11, 12, 13]. 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 [28, 29, 30].

Previous dynamic programming-based approaches have focused on treating gap penalties, rather than position-to-position scores, as objective functions. In parametric sequence alignment [16], 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, [16] finds all optimal alignments within that range. In an attempt to do away with gap penalties entirely, [15] and [17] 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.

2.2 Preliminary Material

2.2.1 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 by go, ge and m.

2.2.2 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 $O(A_n B_n)$ time, where A_n and B_n are the lengths of A and B [31]. The recurrence relations defining the optimal value of an alignment with affine gap weights are

$$\begin{aligned} F_{i,j} &= \max(F_{i-1,j} - ge, V_{i-1,j} - go - ge), \\ E_{i,j} &= \max(E_{i,j-1} - ge, V_{i,j-1} - go - ge), \\ G_{i,j} &= \max(V_{i-1,j-1} + m_{i-1,j-1}, F_{i,j}, E_{i,j}), \\ V_{i,j} &= \max(E_{i,j}, F_{i,j}, G_{i,j}), \end{aligned} \tag{2.1}$$

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 [32] for a more detailed discussion.)

2.2.3 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 $\{\vec{s_1}, \ldots, \vec{s_n}\}$ to an optimization problem, and let f_1 and f_2 be two objective functions. Let $f_1(s_i)$ be the value of objective f_1 for solution s_i. 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 solution dominates the other. We will use the notation $s_1 \succ s_2$ to indicate that s_1 dominates s_2. Over all feasible solutions optimizing f_1 and f_2, we can construct a set of values such that no solution dominates solution 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.

2.3 Pareto Optimal Sequence Alignment

This chapter presents techniques for 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 1 *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 1 *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 2.1, 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

$$\begin{aligned}
\mathcal{F}_{i,j} &= \mathrm{PF}(\{\mathcal{F}_{i-1,j} - \vec{ge}\} \cup \{\mathcal{V}_{i-1,j} - \vec{go} - \vec{ge}\}), \\
\mathcal{E}_{i,j} &= \mathrm{PF}(\{\mathcal{E}_{i,j-1} - \vec{ge}\} \cup \{\mathcal{V}_{i,j-1} - \vec{go} - \vec{ge}\}), \\
\mathcal{G}_{i,j} &= \mathrm{PF}(\{\mathcal{V}_{i-1,j-1} + \vec{\mathbf{m}}_{i-1,j-1}\} \cup \mathcal{F}_{i,j} \cup \mathcal{E}_{i,j}\}), \\
\mathcal{V}_{i,j} &= \mathrm{PF}(\mathcal{E}_{i,j} \cup \mathcal{F}_{i,j} \cup \mathcal{G}_{i,j}),
\end{aligned} \tag{2.2}$$

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 $\mathcal{G}_{i,j}$.

2.3.1 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 improved techniques for eliminating dominated solutions. The first of these techniques uses multi-key sorting and the second uses a tree structure.

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 2.2). 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 $\mathcal{E}_{i,j} = \mathrm{PF}(\{\mathcal{E}_{i,j-1} - \vec{ge}\} \cup \{\mathcal{V}_{i,j-1} - \vec{go} - \vec{ge}\})$, 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.

Dominance Trees

The Dominance tree method described in [29] 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 [11] and Section 2.4.6.

Table 2.1: Notation used in profile scoring function definitions.

Symbol	Description	Definition
$\vec{f}_k$	Amino acid frequencies at position k	
$\vec{p}_k$	Estimated amino acid probabilities at position k	
$\vec{ss}_k$	Predicted secondary structure at position k	
$x \bullet y$	Dot product of x and y	$\sum_a x_a y_a$
$\text{Avg}(x, y)$	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.

2.3.2 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 2.2, the entire frontier needs to be calculated and stored, 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 2.3.3), 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.

2.3.3 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.

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 $[0, 1]$.

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 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.

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 2.4.4) over all proteins in the ce_ref [33] 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 $\forall w_{x_v} : w_{x_v} = 1/k$, this is equivalent to the Unit-space method.

2.4 Methods

2.4.1 Data

To test the effectiveness of various objective combinations for Pareto optimal sequence alignments, we use the ce_ref set of proteins [33]. 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 [34] and use these as our gold standard. (Note that this also makes our results not directly comparable to [9].) 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 [10]. RV11 is one of the most informative datasets [35] 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 [36]. 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.

2.4.2 Profile Construction

Many objective functions require profiles as input, and these are generated using PSI-BLAST version 2.2.12 with the options *-j 5 -e 0.1* (these are the same as specified in [9]). 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 form 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.

2.4.3 Objective Functions

We include eleven objective functions in our study. Nine objectives correspond to the profile-based objective functions from Edgar and Sjölander [9], while the remaining two are the Picasso [37] objective, and SS_dotp. 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 2.1, with the formulas themselves located in Table 2.2.

2.4.4 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) with the much more complicated Cline-shift method [38]. 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 2.2, 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.

2.4.5 Gap Parameter Optimization

Accurate alignments require proper gap-open and gap-extension penalties, which must be conditioned both on the objectives used and on the 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

Table 2.2: Objective match scores, gap parameters and definitions

Name	Go	Ge	Score	Definition
Picasso	9.61	1.92	0.728	$\vec{f_1} \bullet \vec{p_2} + \vec{f_2} \bullet \vec{p_1}$
Correlf	1.30	0.10	0.725	$R(\vec{f_1}, \vec{f_2})$
Rankp	0.78	0.11	0.712	$R(\sigma(\vec{p_1}), \sigma(\vec{p_2}))$
Rankf	0.52	0.08	0.711	$R(\sigma(\vec{f_1}), \sigma(\vec{f_2}))$
Ylf	0.27	0.07	0.702	$(1 - D^{JS}(\vec{p_1}, \vec{p_2}))\times$ $(1 + D^{JS}(\text{Avg}(\vec{p_1}, \vec{p_2}), p^0)$
Fdotf	0.28	0.01	0.701	$\vec{f_1} \bullet \vec{f_2}$
Yldf	0.27	0.03	0.700	$1 - D^{JS}(\vec{f_1}, \vec{f_2})$
Correlp	1.20	0.07	0.693	$R(\vec{f_1}, \vec{f_2})$
Fdotp	0.20	0.03	0.656	$\vec{f_1} \bullet \vec{p_2}$
Ref	1.50	0.40	0.656	$H^S(\vec{f_1}, \vec{f_2})$
SS_dotp	2.11	0.09	0.585	$\vec{ss}_1 \bullet \vec{ss}_2$

Abbreviations used in this table: Go: Gap open cost Ge: Gap extension cost. The Score column shows average match scores over proteins in ce_ref. See Table 2.1 for more information on objective definitions.

400 points, 20 on a 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 [9] except that we use a larger grid and perform global alignments.

2.4.6 Comparison Algorithms

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

Table 2.3: Speed of PF operation methods 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

All values are in seconds and represent 100 alignments.

dividing by the maximum value seen for each objective, and the weights are scaled so that they sum to 1. Objective 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 2.1. 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 2.4.5). 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 2.5.

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* [11], though we replace the $O(N^3)$ original dominance-rank assignment algorithm with a considerably faster algorithm that is $O(N^2)$ in the worst case [29]. 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 [12]. 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

Table 2.4: 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
SS Pc	0.761	0.687	0.801	0.728
SS Cf	0.759	0.686	0.803	0.726
Pc Cp	0.756	0.719	0.757	0.728
Pc Rp	0.753	0.720	0.759	0.728
Cf Cp	0.751	0.715	0.733	0.726
SS Pc Cp	0.770	0.690	0.855	0.728
SS Pc Rp	0.767	0.691	0.861	0.728
SS Pc Cf	0.767	0.691	0.862	0.728
SS Cp Rf	0.767	0.690	0.891	0.712
SS Cf Cp	0.766	0.690	0.854	0.726
SS Pc Cp Rp	0.772	0.691	0.867	0.728
SS Pc Cp Rf	0.771	0.693	0.871	0.728
SS Pc Cf Cp	0.771	0.692	0.855	0.728
SS Pc Cf Rp	0.771	0.692	0.874	0.728
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 on ce_ref.

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 [11] and [12] 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

Table 2.5: Gap parameters and weights on objectives for linear combinations

Objective combination	Go	Ge	α	β	γ	δ
SS Pc	1.95	0.18	0.1	0.9	NA	NA
SS Cf	0.71	0.18	0.2	0.8	NA	NA
Pc Cp	0.58	0.19	0.9	0.1	NA	NA
Pc Rp	0.77	0.19	0.9	0.1	NA	NA
Cf Cp	1.70	0.10	0.5	0.5	NA	NA
SS Pc Cp	0.53	0.18	0.1	0.7	0.2	NA
SS Pc Rp	0.89	0.18	0.1	0.6	0.3	NA
SS Pc Cf	1.42	0.18	0.0	0.3	0.7	NA
SS Cp Rf	0.89	0.18	0.0	0.3	0.7	NA
SS Cf Cp	1.60	0.18	0.0	0.3	0.7	NA
SS Pc Cp Rp	0.89	0.18	0.1	0.6	0.0	0.3
Pc Cf Cp Rp	0.96	0.19	0.6	0.1	0.1	0.2
SS Pc Cf Rp	1.42	0.18	0.0	0.3	0.7	0.0
SS Cf Cp Rp	0.89	0.18	0.3	0.3	0.2	0.2
SS Pc Cp Rf	0.71	0.18	0.1	0.6	0.2	0.1

Abbreviations for objectives: SS: SS_dotp Pc: Picasso Cf: Correlf Cp: Correlp Rf: Rankf Rp: Rankp Description of columns: Go: The penalty for opening a gap. Ge: The penalty for extending a gap. $\alpha, \beta, \gamma, \delta$: The weights on the first, second, third and fourth objectives, respectively. NA: Not applicable: not enough objectives in the combination to require this weight.

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).

2.5 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 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. All experiments were performed on a cluster of 2,186 quad-core 2.8 GHz Intel Xeon X5560 processors.

2.5.1 Single Vs Pareto Optimal Multiple Objectives

We generated Pareto optimal pairwise sequence alignments for all combinations of two, three and four objectives, and evaluate 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 2.4. 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 2.5.3.

Comparing the best alignments generated by the multi-objective approaches with the alignments of the best corresponding single objectives (Table 2.4) 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. Comparing the scores of the alignments from the best corresponding single objectives with the scores of the best alignments generated by the multi-objective approaches, the distributions of the scores are significantly different at $p < 0.1$ for two objectives, $p < 0.01$ for three objectives, and $p < 0.01$ for four objectives.

Examining the fraction improved (FIM) column of Table 2.4, 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 2.1 shows the performance of combinations of four objectives from Table 2.4 relative to their corresponding objectives. The plots in Figure 2.1 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 are 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

Table 2.6: 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
SS Pc	0.730/0.405	0.724/0.400	0.730/0.405
SS Cf	0.723/0.393	0.728/0.395	0.725/0.386
Pc Cp	0.729/0.369	0.722/0.350	0.729/0.371
Pc Rp	0.728/0.362	0.725/0.395	0.728/0.364
Cf Cp	0.722/0.362	0.723/0.354	0.722/0.359
SS Pc Cp	0.729/0.381	0.726/0.372	0.728/0.384
SS Pc Rp	0.730/0.430	0.728/0.403	0.730/0.425
SS Pc Cf	0.733/0.427	0.734/0.417	0.733/0.427
SS Cp Rf	0.726/0.446	0.726/0.437	0.726/0.447
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 ce_ref

frontier frequently outperforms the corresponding single objective.

One interesting aspect of Figure 2.1 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 that of 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.

Table 2.7: 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
SS Pc	0.717/0.250	0.732/0.070	SS Pc Cp	0.727/0.372	0.742/0.297
SS Cf	0.719/0.163	0.729/0.055	SS Pc Rp	0.724/0.366	0.744/0.416
Pc Cp	0.724/0.313	0.740/0.256	SS Pc Cf	0.722/0.366	0.742/0.406
Pc Rp	0.724/0.253	0.741/0.389	SS Cp Rf	0.713/0.417	0.729/0.338
Cf Cp	0.723/0.337	0.735/0.264	SS Cf Cp	0.718/0.284	0.738/0.307
SS Pc Cp Rp	0.724/0.366	0.749/0.484			
SS Pc Cp Rf	0.727/0.381	0.748/0.527			
SS Pc Cf Cp	0.724/0.378	0.748/0.504			
SS Pc Cf Rp	0.722/0.366	0.749/0.566			
SS Cf Cp Rp	0.720/0.367	0.745/0.506			

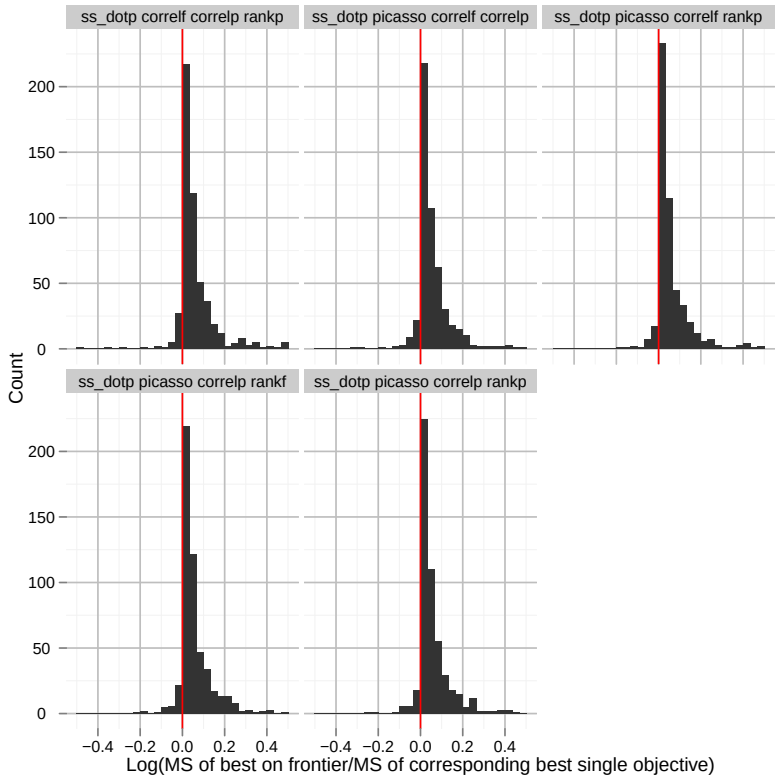
Abbreviations for objectives: SS: SS_dotp Pc: Picasso Cf: Correlf Cp: Correlp Rf: Rankf Rp: Rankp Description of columns: Linear: A linear combination of objectives. EA: Evolutionary algorithm.
MS: Match score. FIM: Fraction improved. Values other than FIM are average match scores over ce_ref

2.5.2 Dominated Solution Elimination

To evaluate dominated solution elimination schemes (Section 2.3.1), 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 2.4–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 2.3.

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

Figure 2.1: Relative improvement of the best solution on Pareto optimal frontiers over the best corresponding single objective when using four objectives.



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.

2.5.3 Coarsening Optimizations

To study the impact of the coarsening methods described in Section 2.3.2 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

Table 2.8: Various performance metrics comparing exact frontiers with Pareto frontiers generated by the Centroid, Cell and Sample coarsening methods under different coarsening parameters using two objectives.

Coarsening method	Coarsening parameter	Alignment time	Percent false positives	Frontier coverage	Best	Mean	BPO
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	0.1	6.596	33.308	53.001	0.727	0.666	0.696
Centroid	0.05	6.367	33.997	54.163	0.728	0.667	0.696
Centroid	0.025	6.326	34.883	55.811	0.730	0.666	0.696
Centroid	0.012	6.655	34.471	58.448	0.730	0.665	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	10	6.479	33.626	57.661	0.730	0.667	0.696
Cell	15	6.645	32.882	59.550	0.731	0.667	0.696
Cell	20	6.845	31.947	61.170	0.731	0.667	0.696
Cell	25	7.041	31.277	62.438	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	0.5	6.782	31.689	61.907	0.730	0.668	0.696
Sample	0.75	7.545	28.783	65.788	0.731	0.668	0.696
Sample	0.8	7.751	28.074	66.635	0.731	0.668	0.696
Sample	0.85	7.977	27.267	67.490	0.731	0.668	0.696
Sample	0.9	8.281	26.161	68.641	0.731	0.668	0.696

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. The row labeled "None" shows results for the exact frontier.

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 the results for two objectives in Table 2.8, for three objectives in Table 2.9 and for four objectives in Table 2.10.

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

Table 2.9: Various performance metrics comparing exact frontiers with Pareto frontiers generated by the Centroid, Cell and Sample coarsening methods under different coarsening parameters using three objectives.

Coarsening method	Coarsening parameter	Alignment time	Percent false positives	Frontier coverage	Best	Mean	BPO
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	0.1	4.897	58.575	31.938	0.817	0.760	0.790
Centroid	0.05	5.313	58.948	33.094	0.822	0.762	0.790
Centroid	0.025	8.831	58.798	34.818	0.825	0.761	0.790
Centroid	0.012	25.108	56.889	38.392	0.828	0.761	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	10	5.566	55.316	39.569	0.828	0.766	0.790
Cell	15	6.332	52.534	42.729	0.829	0.766	0.790
Cell	20	7.127	50.574	44.858	0.829	0.766	0.790
Cell	25	7.991	48.835	46.689	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	0.5	5.319	55.219	39.753	0.827	0.769	0.790
Sample	0.75	6.348	51.408	43.934	0.828	0.769	0.790
Sample	0.8	6.692	50.351	45.001	0.829	0.770	0.790
Sample	0.85	7.152	48.901	46.330	0.829	0.769	0.790
Sample	0.9	7.944	46.789	48.243	0.829	0.769	0.790

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. The row labeled "None" shows results for the exact frontier.

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 2.2 and 2.3. Each point in Figure 2.2 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 2.3 plots the match score of the best solution on the approximate frontier relative to that of the best solution on the exact

Table 2.10: Various performance metrics comparing exact frontiers with Pareto frontiers generated by the Centroid, Cell and Sample coarsening methods under different coarsening parameters using four objectives.

Coarsening method	Coarsening parameter	Alignment time	Percent false positives	Frontier coverage	Best	Mean	BPO
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	0.1	3.944	61.691	31.733	0.852	0.796	0.822
Centroid	0.05	7.675	61.828	32.691	0.858	0.798	0.822
Centroid	0.025	27.578	60.969	34.764	0.861	0.798	0.822
Centroid	0.012	90.810	56.305	40.365	0.864	0.799	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	10	5.157	54.074	42.267	0.863	0.804	0.822
Cell	15	6.920	50.028	46.445	0.865	0.804	0.822
Cell	20	9.596	46.707	49.667	0.866	0.804	0.822
Cell	25	13.299	43.284	53.104	0.866	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	0.5	3.992	57.406	38.680	0.862	0.807	0.822
Sample	0.75	4.923	52.885	43.500	0.864	0.808	0.822
Sample	0.8	5.275	51.461	44.846	0.864	0.807	0.822
Sample	0.85	5.803	49.634	46.578	0.865	0.807	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. The row labeled "None" shows results for the exact frontier.

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 2.2 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 overlap between the approximate and exact frontiers is small (see Tables 2.8, 2.9 and 2.10), 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.

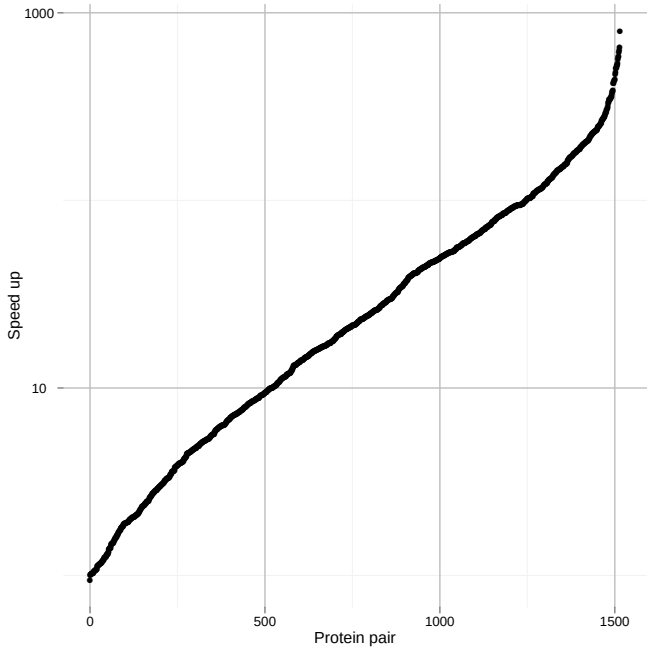


Figure 2.2: Four objective coarsening algorithm run-time performance.

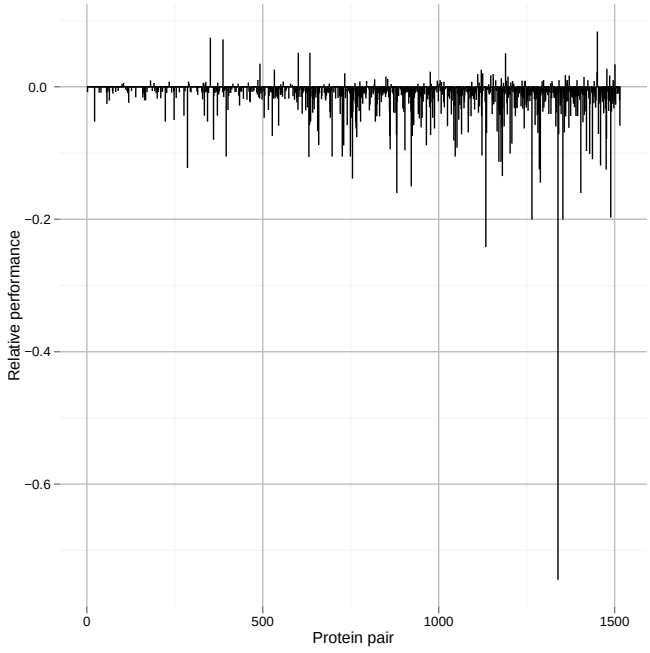


Figure 2.3: Four objective coarsening algorithm alignment quality.

Table 2.11: Various performance metrics describing the runtime and quality of the solutions generated by the evolutionary algorithm.

NOB	Time	Percent false positives	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. NOB: Number of objectives. 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

2.5.4 Solution Selection

The results in Table 2.4 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 2.6 shows the performance of the various solution selection schemes described in Section 2.3.3 over the 588 protein pairs from the ce_ref set. Comparing Table 2.6 with Table 2.4, 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.

2.5.5 Comparisons with other Algorithms

Table 2.7 shows the performance of linear combinations of objectives and the best solution generated by the evolutionary algorithm (see Section 2.4.6). 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 2.7) with the selection methods (Table 2.6), 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 2.2 and 2.7 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 2.7) with the "Best" column in Table 2.4, 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 2.5.4).

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 long run times. With four objectives, the evolutionary approach produces around 4% coverage in around 1820 seconds (see Table 2.11) which is considerably more than the 241 seconds required for the dynamic programming approach to generate the exact Pareto optimal frontier. 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 (Table 2.8), three (Table 2.9) and four objectives (Table 2.10), while the evolutionary algorithm generates 3.15%, 3.17% and 4.12% (Table 2.11). An interesting observation from the FIM values in Tables 2.4, 2.6 and 2.7 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.

Performance Summary

Table 2.12 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 [39] 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.

2.6 Discussion

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 2.3), 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 2.3, 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.

Table 2.12: 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 ce_ref; 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

Chapter 3

Protein Structure Assembly

New fold prediction techniques assemble a structure for a query in the absence of a template with more than 25% sequence identity to the query. These techniques are either computational biology approaches based on modeling physiochemical forces, or knowledge-based approaches based on fragment assembly. Fragment assembly methods attempt to build a structure from pieces of other known structures. The idea behind this is that protein fragments can constrain their own structure, but not be overly influenced by the protein in which they are contained. This implies that they can be taken from a template structure and used to build a different structure for a query. There are three primary components to a fragment assembly procedure: fragment selection, fragment assembly, and structure evaluation.

Proper selection of a set of fragments to use for fragment assembly is crucial, as the fragments will greatly influence the quality of the resulting prediction. With too few fragments, or a set of fragments lacking sufficient diversity, an accurate structure for the query cannot be constructed. With too many fragments, the number of possible combinations for those fragments becomes too large to be computationally tractable. The length of the fragments used is also important, as longer fragments will constrain a structure more than shorter fragments. Even with the limited number of known protein structures, there are far too many possible protein fragments to use all of them in a fragment assembly-based prediction. To limit the number of fragments that must be considered, fragment libraries based on structural clusters of protein fragments have been constructed. Specific fragments are typically chosen on the basis of evolutionary

sequence similarity to the query.

Given a set of fragments for a query, a structure is built from a subset of those fragments. The structure for a query starts as an extended chain of amino acids. The coordinates of the query are replaced with the coordinates from one or more fragments, and the result is evaluated using a scoring function. Based on this evaluation, the new structure can be kept or rejected. This process is repeated until some stopping criteria are met. To assemble the best possible structure, previous assembly-based approaches use an algorithm involving a stochastic search (e.g. simulated annealing [40], genetic algorithms [41], or conformational space annealing [42]). However, these search algorithms cannot guarantee either locally or globally optimal results.

In this chapter we present the relative performance of deterministic and stochastic techniques for optimizing the assembly of protein fragments into a complete structure. The new algorithms presented below are inspired by techniques originally developed in the context of graph partitioning [43]. The Greedy approach examines all possible fragment insertions at a given point and chooses the best one available. The Hill-climbing algorithm follows a similar strategy but allows for moves that locally reduce the score, provided that they lead to a better global score.

Several variables affect the performance of optimization algorithms in the context of fragment-based *ab initio* structure prediction, such as the number of fragments available to the optimizer, the length of the fragments, if they should be multiple sizes at different stages [40] or all different sizes used together [41], and other parameters specific to the optimizer. Consequently, we vary the fragment length and the number of fragments per position when comparing the performance of our optimization algorithms to that of a tuned simulated annealing approach. Our experiments test these algorithms on a diverse set of 276 protein domains derived from the Structural Classification of Proteins (SCOP) version 1.69 [44]. The results of these experiments show that the Hill-climbing-based approaches are very effective in producing high quality structures in a moderate amount of time, and that they generally outperform simulated annealing. On the average, Hill-climbing is able to produce structures that are 6% to 20% better as measured by the root mean square deviation (RMSD) between the predicted and actual structures. Furthermore, the relative advantage of Hill-climbing-based approaches improves with the length of the proteins.

3.1 Related Research

Historically, the random nature of evolutionary and energetic processes have motivated the use of stochastic algorithms for optimizing fragment placement. In addition to simulated annealing (see Section 3.2.5), genetic algorithms have been used for some time in structure prediction [41, 45, 46]. The basic idea behind genetic optimization is to allow a population of candidates to evolve randomly in the hopes that at least one will reach an optimal state. For structure prediction, a set of structures constitutes a population, and the objective function directs its evolution. Initially, such algorithms made simple modifications to individual dihedral angles to model mutations, or swapped sets of dihedral angles between different structures to model recombinations. As the field has progressed, more complicated conformational changes have allowed for better optimization of the objective function. Although the dihedral angles must still be the eventual target of such operations, the optimization can replace multiple fragments simultaneously, make rigid body movements of disconnected portions and move side-chains independently from the backbone [45].

Another approach, called conformational space annealing [42], incorporates aspects of both simulated annealing and genetic algorithms. A set of structures provides a *bank*, which is similar to a population in genetic algorithms. As the optimization progresses, structures from this bank are modified or discarded based on the value of the objective function and an annealing parameter. As in simulated annealing, this parameter is slowly changed to focus on better values of the objective function. This allows the optimization to maintain diversity in the bank in early stages while finding good values of the objective function in later stages.

An interesting similarity between both simulated annealing-based approaches and genetic algorithms is that the objective function can be directly linked to the optimization algorithm. Genetic algorithms can modify the rate at which certain modifications are made based on previous results [45], and simulated annealing can incorporate different terms of the objective function at different temperatures [47], and/or modify its temperature based on the current value of that function. In the algorithms presented below, the optimization technique is decoupled from the objective function.

Recently, greedy techniques (which make the locally optimal choice at each stage)

have been applied to determine a set of representative fragments for use in decoy structure construction [48, 49], and to reconstruct a native protein fold given such a set of representative fragments [50, 51]. The greedy approaches used for both these problems traverse the query sequence in order, inserting the best fragment found for each position. As an extension, the algorithms build multiple structures simultaneously in the search for a better structure. While such approaches have the ability to avoid local minima, they lack an explicit notion of hill-climbing.

3.2 Methods

3.2.1 Data

Optimization algorithm performance is evaluated using a set of proteins derived from SCOP 1.69 [44]. Starting from the set of domains in SCOP, we removed all membrane and cell surface proteins, then used tools from Astral [52] to construct a set of proteins with less than 25% sequence identity. This set is further reduced by keeping only the structures that are determined by X-ray crystallography, filtering out any proteins with a resolution worse than 2.5Å, and removing any proteins with a $C_\alpha - C_\alpha$ distance greater than 3.8Å times their sequential separation.[1]

Table 3.1: Number of sequences at various length intervals and SCOP class.

	Sequence Length			
SCOP Class	< 100	100–200	> 200	total
alpha	23	40	6	69
beta	23	27	18	69
alpha/beta	4	26	39	69
alpha+beta	15	36	17	69

The above steps produce a set of 2817 proteins. From these, we selected a subset of 276 proteins for use in evaluating the performance of the various optimization algorithms while the remaining 2541 sequences serve as the database of structural fragments. The test sequences, whose characteristics are summarized in Table 3.1, are selected to have diverse lengths and secondary structure compositions.

[1] No bond lengths are modified to fit this constraint; proteins not satisfying it are simply removed from consideration.

3.2.2 Neighbor Lists

The number of possible configurations of protein fragments grows exponentially with the number of fragments and the length of the protein being constructed (the *query*). As the length of the protein cannot change, to keep the problem tractable fragment-based *ab initio* structure prediction approaches restrict the number of structural fragments considered for each amino acid of the query sequence. In evaluating the various optimization algorithms we developed, we followed a methodology similar in spirit to that used by Rosetta [40] for identifying these structural fragments.

Consider a query sequence A of length A_n. For each amino acid $a_i \in A$, we identify a list ($\mathcal{L}_i$) of n structural fragments by comparing the query with the sequences in the training set. We refer to a contiguous subsequence of length k as a *k-mer*. To construct $\mathcal{L}_i$, we compare the k-mer of A starting at position i ($0 \leq i \leq A_n - k + 1$) with all k-mers in the training set. The n members of $\mathcal{L}_i$ have the n highest profile-based scores with the query sequence's k-mer. We will refer to the list $\mathcal{L}_i$ as the *neighbor list* for position i.

Regarding the length of the fragments in neighbor lists, we use fragments of a single length as well as fragments of different lengths. In the latter case we consider two different approaches for incorporating fragments of different lengths. The first, referred to as *scan*, uses the fragment lengths in decreasing order. For example, if the neighbor lists contain structural fragments of length three, six, and nine, the algorithm optimizes the structure using only fragments of length nine, then fragments of length six, and finally fragments of length three. Each one of these optimization phases terminates when the algorithm has either reached a local optimum or performed a predetermined number of iterations, and the resulting structure becomes the input to the subsequent optimization phase. The second approach for combining different length fragments is referred to as *pool*, and it optimizes the structure once, selecting fragments from any available length. The *fragment selection scheme* dictates which size fragment to use, and when.

Sequence Profiles

The comparisons between the query and the training sequences use evolutionary

profiles generated by PSI-BLAST [53]. The profile of a sequence A of length A_n is represented by two $A_n \times 20$ matrices. The first is its position-specific scoring matrix PSSM_A that is computed directly by PSI-BLAST. The rows of this matrix correspond to the various positions in A, while the columns correspond to the 20 distinct amino acids. The second matrix is its position-specific *frequency* matrix PSFM_A that contains the frequencies used by PSI-BLAST to derive PSSM_A. These frequencies (also referred to as *target frequencies* [54]) contain both the sequence-weighted observed frequencies (also referred to as *effective frequencies* [54]) and the BLOSUM62 derived pseudocounts [53, 55]. For each row of a PSFM, the frequencies are scaled so that they sum to one. In the cases where PSI-BLAST could not produce meaningful alignments for a given position of A, the corresponding rows of the two matrices are taken from the scores and frequencies of BLOSUM62.

For our study, we use the version of the PSI-BLAST algorithm available in NCBI's blast release 2.2.10 to generate profiles for both the test and training sequences. These profiles are derived from the multiple sequence alignment constructed after five iterations using an e value of 0.1. The PSI-BLAST search is performed against NCBI's nr database downloaded in November of 2004, which contained 2,171,938 sequences.

Profile-to-Profile Scoring Method

The similarity score between a k-mer from the query sequence and a k-mer from a sequence in the training set is the score of the ungapped alignment between the two k-mers. Many different schemes have been developed for determining the similarity between profiles, combining information from the original sequence, position-specific scoring matrix, or position-specific target and/or effective frequencies [54, 56, 57]. In our work we use a scheme that is derived from the Picasso [37, 54] function, used in developing effective remote homology prediction and fold recognition algorithms [58]. Specifically, the similarity score between the ith position of protein A's profile, and the jth position of protein B's profile is given by

$$\begin{aligned} S_{A,B}(i,j) = & \sum_{a=1}^{20} \text{PSFM}_A(i,a)\,\text{PSSM}_B(j,a) \; + \\ & \sum_{a=1}^{20} \text{PSFM}_B(j,a)\,\text{PSSM}_A(i,a), \end{aligned} \tag{3.1}$$

where $\text{PSFM}_A(i, a)$ and $\text{PSSM}_A(i, a)$ are the values corresponding to the ath amino acid at the ith position of A's position-specific scoring and frequency matrices, and where $\text{PSFM}_B(j, a)$ and $\text{PSSM}_B(j, a)$ are the values corresponding to the ath amino acid at the jth position of B's position-specific scoring and frequency matrices.

Equation 3.1 determines the similarity between two profile positions by weighting the position-specific scores of the first sequence according to the frequency at which the corresponding amino acid occurs in the second sequence's profile. The key difference between Equation 3.1 and Picasso [54] is that our measure uses the target frequencies, whereas Picasso is based on effective frequencies.

3.2.3 Protein Structure Representation

We consider only the positions of the C_α atoms, and we use a vector representation of the protein in lieu of ϕ and ψ backbone angles. Our protein construction approach uses the coordinates of the atoms in each fragment, rotated and translated into the reference frame of the working structure.

3.2.4 Scoring Function

As we are developing new optimization techniques, we use the RMSD between the predicted and native structure of a protein as the scoring function to be optimized. Although such a function cannot serve as a predictive measure, using this as a scoring function allows for a clear distinction between the optimization process and the scoring function. In effect, we assume an ideal scoring function in order to test the optimization techniques.

3.2.5 Optimization Algorithms

We compare the performance of three different optimization algorithms in the context of fragment-assembly based approaches for *ab initio* structure predictions. One of these algorithms, simulated annealing [59], is a widely used method to solve such problems, whereas the other two algorithms, Greedy and Hill-climbing, are newly developed.

The key operation in all three of these algorithms is the replacement of a k-mer starting at a particular position i with that of a structural fragment from a neighbor

list. We will refer to this operation as a *move*. A move is considered valid if it does not create any steric conflicts. A structure is considered to have a steric conflict if it contains a pair of C_α atoms within 2.5Å of one another. For each valid move, its *gain* is defined as the difference in the value of the scoring function between the working structure and the native structure of the protein.

Simulated Annealing (SA)

Simulated annealing [59] is a generalization of the Monte Carlo [60] method for discrete optimization problems. This optimization approach is designed to mimic the process by which a material such as metal or glass cools. At high temperatures, the atoms of a metal can adopt configurations not available to them at lower temperatures—e.g., a metal can be a liquid rather than a solid. As the system cools, the atoms arrange themselves into more stable states, forming a stronger substance.

The simulated annealing (SA) algorithm proceeds in a series of discrete steps. In each step it randomly selects a valid move and performs it (i.e., inserts the selected fragment into the structure). If the move improves the quality, then the move is accepted. If it degrades the quality, then the move will still be accepted with probability p, where

$$p = e^{\left(\frac{S_{old}-S_{new}}{T}\right)}, \tag{3.2}$$

T is the current temperature of the system, S_{old} is the score of the last state, and S_{new} is the score of current state. From Equation 3.2 we see that the likelihood of accepting a bad move is inversely related to the temperature combined with how much worse the new structure is than the old structure. In other words, the optimizer is more likely to accept a very bad move if the temperature is high than if the temperature is low.

The algorithm begins with a high system temperature which progressively decreases according to an *annealing schedule*. As the optimization must use finite steps, the cooling of the system cannot be continuous, but the annealing schedule can be modified to increase its smoothness. The annealing schedule depends on a combination of the number of steps and the total number of allowed moves over these steps. Our implementation of simulated annealing uses an annealing schedule that linearly decreases the temperature of the system to zero over a fixed number of cycles. Due to its stochastic

nature, picking the best result over a number of iterations of the complete schedule improves performance.

Simulated annealing is a highly tunable optimization framework. The starting temperature and the annealing schedule can be varied, and the performance of the algorithm depends greatly on these parameters. Section 3.3.2 describes how we arrive at the values for the parameters of SA as implemented in this study.

The Greedy Algorithm

One of the characteristics of the simulated annealing algorithm is that it considers moves at random, irrespective of their gains. In contrast, the Greedy algorithm presented here selects maximum gain moves. The algorithm consists of two phases. In the first phase, called *initial structure generation*, the protein is a fully-extended chain and the algorithm attempts to make a valid move at each position. This is achieved by scoring all neighbors in each neighbor list and inserting the neighbor with the highest gain from each list. If some positions have no valid moves on the first pass, the algorithm attempts to make moves at these positions after trying all positions once. This ensures that the algorithm makes moves at nearly every position, and also provides a good starting point.

In the second phase, called *progressive refinement*, the algorithm repeatedly finds the maximum gain move over all positions, and if this move is valid and the gain is positive, the algorithm makes the move. The progressive refinement phase terminates upon failing to find any move to make. The Greedy algorithm is guaranteed to terminate at either a local or global optimum.

Hill-Climbing (HC)

The Hill-climbing algorithm was developed to allow the Greedy algorithm to "climb" out of locally optimal solutions. The key idea behind Hill-climbing is to continue performing valid moves after achieving a local optimum in the hope of finding a better solution.

The Hill-climbing algorithm works as follows. The algorithm begins by applying the Greedy algorithm in order to reach a local optimum. At this point, it begins a sequence of iterations consisting of a *hill-climbing* phase, followed by a progressive refinement phase. In the hill-climbing phase, the algorithm performs a series of moves, each time

selecting the maximum gain valid move irrespective of whether that gain is positive or negative. If at any point during this series of moves, the working structure achieves a score that is better than that of the structure at the beginning of the hill-climbing phase, this phase terminates and the algorithm enters the progressive refinement phase. The above sequence of iterations terminates when the hill-climbing phase is unable to produce a better structure after successively performing a maximum scoring move at each position.

Since the hill-climbing phase starts at a local optimum, its initial set of moves will lead to a structure whose quality, as measured by the scoring function, is worse than that at the beginning of the hill-climbing phase. However, subsequent moves can potentially lead to improvements that outweigh the initial quality degradation, allowing the algorithm to climb out of locally optimal solutions.

Move locking As Hill-climbing allows negative gain moves, the algorithm can oscillate between a local optimum and a non-optimal solution. To ensure this does not happen, a *lock* is placed on each successful move to prevent it from being made again within the same phase. All locks are cleared at the end of a hill-climbing phase.

We present two different locking methods. The first, referred to as *fine-grain locking*, locks only the move made. The algorithm can subsequently select a different neighbor for insertion at this position. The second, referred to as *coarse-grain locking*, locks the position of the query sequence, preventing any further insertions at that position. In the case of pooling different size fragments, coarse-grain locking locks moves of all sizes.

Since fine-grain locking is less restrictive, we expect it to lead to better quality solutions. The advantage of coarse-grain locking is that each successive fragment insertion significantly reduces the set of fragments that need to be considered for future insertions, leading to a faster optimization.

Efficient Checking of Steric Conflicts

The Greedy and Hill-climbing algorithms need to evaluate the validity of every available move after every insertion, as each insertion can introduce new steric conflicts. In an attempt to reduce the time required for this process, we have developed an efficient formulation for validity checking. Recall from Section 3.2.5 that a valid move brings

no two C_α atoms within 2.5Å of each other. To quickly determine if this proximity constraint holds, we impose a three-dimensional grid over the structure being built with boxes 2.5Å on each side. As each move is made, its atoms are added to the grid, and for each addition only the 26 adjacent boxes are checked for proximity constraint violations. This limits the number of distances that must be computed.

We further decrease the required time by sequentially checking neighbors at each position down the amino acid chain. All atoms upstream of the insertion point must be internally valid, as they have previously passed proximity checks. Thus we only need to examine those atoms at, or downstream from, the insertion.

3.3 Results

3.3.1 Performance of the Greedy and Hill-climbing Algorithms

To compare the effectiveness of the Greedy and Hill-climbing optimization techniques, Table 3.2 shows results for the Greedy and Hill-climbing optimization techniques using k-mer sizes of 9, 6, and 3 individually, as well as using the scan and pool techniques to combine them. Average times are also reported for each of these five fragment selection schemes. All times are from machines running dual core 2 Ghz AMD Opteron 270 processors with 4 GB of system memory.

Examining Table 3.2, we see that the Hill-climbing algorithm consistently outperforms the Greedy algorithm. As Hill-climbing includes running Greedy to convergence, the result is not surprising, and neither is the increased run-time that Hill-climbing requires. Both schemes seem to take advantage of the increased flexibility of smaller fragments and greater numbers of fragments per position. For example, on the average the 3-mer results are 9.4%, 12.0%, and 8.5% better than the corresponding 9-mer results for Greedy, Hill-climbing coarse (HC_c) and Hill-climbing fine (HC_f), respectively. Similarly, increasing the neighbor lists from 25 to 100 yields a 23.1%, 31.6%, and 43.6% improvement for Greedy, HC_c, and HC_f, respectively. These results also show that these algorithms are progressively more powerful as the size of the overall search space increases.

With respect to locking, a less restrictive fine-grained approach generally yields better results than a coarse-grained scheme. For example, averaging over all experiments,

Table 3.2: Average values over 276 proteins optimized using Hill-climbing and different locking schemes. Times are in seconds and scores are in Å. Lower is better in both cases.

	$n = 25$		$n = 50$		$n = 75$		$n = 100$	
	Score	Time	Score	Time	Score	Time	Score	Time
Greedy								
$k = 9$	9.07	11	8.20	14	7.77	18	7.38	22
$k = 6$	8.76	12	7.98	17	7.50	22	7.21	27
$k = 3$	8.20	15	7.51	22	7.08	30	6.80	39
Scan	7.19	33	6.49	40	6.00	48	5.79	56
Pool	7.06	41	6.34	58	5.94	76	5.57	97
Hill-climbing coarse (HC_c)								
$k = 9$	6.70	49	5.99	98	5.54	143	5.29	226
$k = 6$	6.46	65	5.67	124	5.23	221	4.93	279
$k = 3$	6.07	76	5.35	182	4.92	313	4.68	433
Scan	5.07	120	4.47	216	4.01	333	3.76	517
Pool	5.06	341	4.33	912	3.96	1588	3.74	1833
Hill-climbing fine (HC_f)								
$k = 9$	5.81	357	4.96	1314	4.53	2656	4.30	4978
$k = 6$	5.67	352	4.76	1417	4.30	3277	3.99	5392
$k = 3$	5.56	390	4.60	1561	4.10	3837	3.87	6369
Scan	4.65	736	3.92	2878	3.37	6237	3.17	10677
Pool	4.30	1997	3.56	7101	3.14	18000	2.87	21746

fine-grained locking yields a 21.2% improvement over coarse-grained locking. However, this increased performance comes at the cost of an average increase in run-time of 1128%.

Comparing the performance of the scan and pooling methods to combine variable length k-mers we see that pool performs consistently better than scan by an average of 4.4%. This improvement also comes at the cost of an average increase in run time of 131.1%. Results from the pool and scan settings clearly indicate that Greedy and HC_c are not as effective at exploring the search space as HC_f.

3.3.2 Comparison with Simulated Annealing

Tuning the Performance of SA

Due to the sensitivity of simulated annealing to specific values for various parameters, we performed a search on a subset of the test proteins in an attempt to maximize the ability of SA to optimize the test structures. Specifically, we attempted to find values for two governing factors: the initial temperature T_0 and the number of moves mo.

To this end, we selected ten medium length proteins of diverse secondary structural classification (see Table 3.3), and optimized them over various initial temperatures. The initial temperature that yielded the best average RMSD was $T_0 = 0.1$ and we use this value in all subsequent experiments.

In addition to an initial temperature, when using simulated annealing one must select an appropriate annealing schedule. Our annealing schedule decreases the temperature linearly over 3500 cycles. The algorithm attempts $mo = \alpha \times A_n \times n$ moves, where α is an empirically determined scaling factor, A_n is the number of amino acids in the query protein, and n is the number of neighbors per position. For the scan and pool techniques (see Section 3.2.2), we allow SA three times the number of attempted moves because the total number of neighbors is that much larger. In order to produce comparable run-times to the Greedy, HC_c and HC_f schemes, α values of 20, 100 and 200 are employed, respectively. Finally, following recent work [47] we allow for a temporary increase in the temperature after 150 consecutively rejected moves.

Table 3.3: Tuning set SCOP classes and lengths

SCOP identifier	Length	SCOP class
d1jiwi_	105	beta
d1kpf__	111	alpha+beta
d2mcm__	112	beta
d1bea__	116	alpha
d1ca1_2	121	beta
d1jiga_	146	alpha
d1nbca_	155	beta
d1yaca_	204	alpha/beta
d1a8d_2	205	beta
d1aoza2	209	beta

Due to its stochastic nature simulated annealing produces a different result each time it is run. We call the number of complete runs r. For both $\alpha = 20$ and $\alpha = 100$ we set r to one and for $\alpha = 200$ we set r to five.

Table 3.4: Average performance over 276 proteins optimized using simulated annealing. Times are in seconds and scores are in Å. Lower is better in both cases.

		$n = 25$		$n = 50$		$n = 75$		$n = 100$	
		Score	Time	Score	Time	Score	Time	Score	Time
$\alpha = 20$ $r = 1$	$k = 9$	7.88	25	6.99	31	6.54	36	6.28	42
	$k = 6$	7.45	25	6.46	30	6.12	36	6.03	42
	$k = 3$	6.78	25	6.01	31	5.87	37	5.81	43
	Scan	6.10	74	5.53	92	5.39	109	5.38	128
	Pool	5.93	75	5.84	94	6.00	112	6.13	132
$\alpha = 100$ $r = 1$	$k = 9$	6.76	52	6.34	81	6.31	112	6.28	145
	$k = 6$	6.31	50	6.14	81	6.18	115	6.26	146
	$k = 3$	6.05	52	6.21	84	6.34	118	6.40	155
	Scan	5.60	147	5.52	240	5.55	341	5.60	438
	Pool	5.99	156	6.23	265	6.34	352	6.38	447
$\alpha = 200$ $r = 5$	$k = 9$	5.78	317	5.74	624	5.78	1000	5.80	1437
	$k = 6$	5.66	327	5.83	653	5.89	981	5.94	1360
	$k = 3$	5.89	336	6.09	674	6.13	1060	6.18	1473
	Scan	5.12	1134	5.11	2174	5.11	3268	5.18	4305
	Pool	5.68	1063	5.86	2164	5.92	3805	5.95	5355

The values of α in the above table scale the number of moves simulated annealing is allowed to make. In our case, the total number of moves is $\alpha \times l \times n$ where l is the length of the protein being optimized and n is the number of neighbors per position. The value of r is the number of independent runs attempted, from which the best possible value is taken.

Divided Results

The simulated annealing results are summarized in Table 3.4. As we see in this table, simulated annealing consistently outperforms the Greedy scheme. Specifically, the average performance of SA with $\alpha = 20$ is 15.1% better, based on the average ratio between RMSDs for the two methods over all fragment selection schemes and all values of n. The superior performance of simulated annealing over Greedy is to be expected, as Greedy lacks any sort of hill-climbing ability, whereas the stochastic nature of simulated annealing allows it a chance of overcoming locally optimal solutions. In contrast, both the fine and coarse-locking versions of Hill-climbing outperform SA. More concretely, on the average HC_c performs 22.8% better than SA with $\alpha = 100$, and HC_f performs 36.6% better than SA with $\alpha = 200, r = 5$. Furthermore, HC_c performs 14% better than SA with $\alpha = 200, r = 5$, while taking an average of 495.5% less time.

Analyzing the results shown in Table 3.4, the performance occasionally decreases as the α value increases. This seemingly strange result comes from the dependence of the cooling process on the number of allowed moves, in which the value of α plays a role. For all entries in Table 3.4 the annealing schedule will cool the system over a fixed

number of steps, but the number of moves made will vary greatly. Thus, in order to keep the cooling of the system linear we vary the number of moves allowed before the system reduces its temperature. As a result, different values of α can lead to different randomly chosen optimization paths.

Comparing the performance of the various optimization schemes with respect to the various fragment selection schemes, we see an interesting trend. The performance of SA deteriorates (by an average of 9.9%) when the different length k-mers are used via the pool method, whereas the performance of HC_f improves (by 4.4% on average).

This behavior results because simulated annealing has a bias towards smaller fragments, as an insertion of a bad 3-mer will degrade the structure less than that of a bad 9-mer, and consequently the likelihood of accepting the former move will be higher (Equation 3.2).

Performance on different length sequences To better understand how the length of the query sequences affects the performance of the different optimization schemes, we divided the 276 proteins of our test set into a set containing the 138 shortest sequences and a set containing the 138 longest sequences. The length of the proteins in the first set ranged from 33 to 140 with an average length of 99.1±27.0, whereas the length of the proteins in the second set ranged from 141 to 759 with an average length of 248.7±114.3. Fig. 3.1 shows the relative improvement of Greedy, HC_c, and HC_f as compared to SA with α equal to 20, 100, and 200, respectively, for these subsets as well as the entire set of proteins.

Even though the overall trends in this figure agree with those observed for all the proteins, a key point is that the relative performance of the various schemes depends on the length of the proteins. For the longer sequences, the improvement of the Hill-climbing scheme over simulated annealing is higher than that achieved for the shorter sequences, whereas the Greedy scheme does relatively worse than simulated annealing as the length of the proteins increases. For example, comparing Greedy and SA for $\alpha = 20$, Greedy performs 12.1% worse for the shortest sequences, as opposed to 13.6% worse for the longest sequences. Comparing HC_c and SA for $\alpha = 100$, HC_c performs 28.4% better for the longest sequences as opposed to 13.1% better for the shortest sequences. Finally, comparing HC_f and SA for $\alpha = 200$ and with r set to five, HC_f is 46.1% better for

the longest sequences, as opposed to 20.6% for the shortest sequences. Since the search space associated with longer sequences is larger, these results suggest that within an extended search space, (i) the hill-climbing ability is important, and (ii) the HC_c and HC_f schemes are more effective in exploring large search spaces than SA.

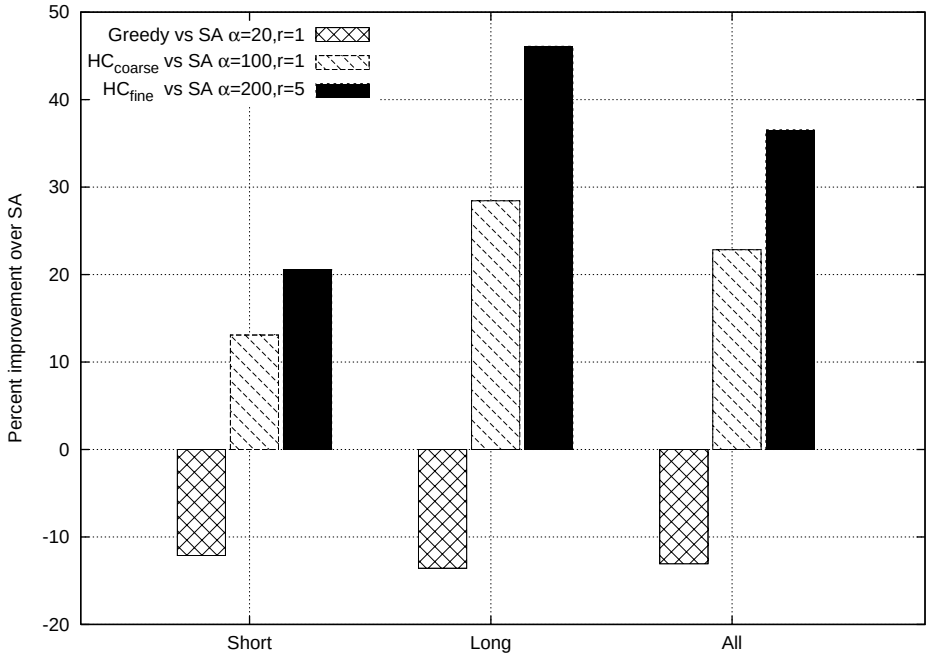


Figure 3.1: Improvement of greedy-based algorithms over simulated annealing, divided by length.

Performance on the different SCOP classes To study how the overall 3D structure of the proteins affects the performance of the different optimization schemes we also divide the proteins in our test set according to their SCOP class (i.e., α, β, α/β, and $\alpha+\beta$). Figure 3.2 shows the relative improvements achieved by Greedy, HC_c, and HC_f over SA for each of these four subsets. Our test set contains the same number of proteins from each SCOP class (Table 3.1).

These results show that the relative performance of the different schemes depends on the overall structure of the proteins being predicted. Even though the relative performance of the various optimization schemes for α, β and α/β proteins shows little

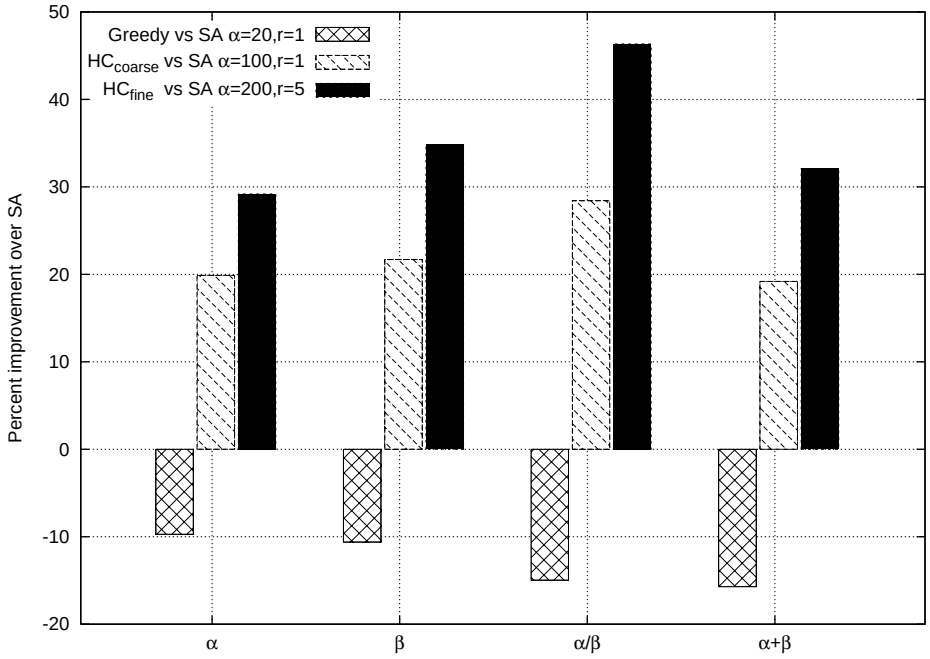


Figure 3.2: Improvement of greedy-based algorithms over simulated annealing, divided by class.

variation, there is a marked difference between the α/β class and the other three. This variation is primarily due to proteins in the α/β class tending to be longer than those in the other classes. In the test set, the average length is 134.71±91.3 for α proteins, 153.2±96.6 for β proteins, 252.3±139.9 for α/β proteins and 155.3±68.6 for $\alpha + \beta$ proteins. As the results in Figure 3.1 show, the hill-climbing schemes show a higher relative improvement on the longer sequences, so it is not surprising that the biggest improvement is in the class with the largest average length.

3.4 Discussion

We present two new techniques for optimizing scoring functions for protein structure prediction. One of these approaches, HC$_c$, using the scan technique, reaches better solutions than simulated annealing in a comparable amount of time. The performance of SA seems to saturate beyond $\alpha = 50$, but HC$_f$ can make use of an increased time

allowance, finding the best solutions of all the examined algorithms. Furthermore, experiments with variations on the number of moves available to the optimizer demonstrate that the Hill-climbing approach makes better use of an expanded search space than simulated annealing. Additionally, simulated annealing requires tuning several parameters, including the total number of moves, the initial temperature, and the annealing schedule. One of the main advantages of using schemes like Greedy and Hill-climbing is that they do not rely on such parameters.

These algorithms can be easily parallelized to further reduce the time required by the optimization process. The rate-limiting step in both the Hill-climbing and Greedy algorithms is determining the valid move that achieves the maximum gain over all positions of the chain. This step can be performed in parallel by having each processor check a subset of the moves and then selecting the best move among the processors with a global reduction operation. This approach will parallelize the most expensive portion of the overall computation and will dramatically reduce the amount of time taken by the algorithm.

In addition to the optimization methods, the objective function being optimized is also critical. Current objective functions [40, 41, 42] do not correlate perfectly with the distance to the native conformation. Consequently, the global optimum solution may not necessarily correspond to the native state. Nevertheless, irrespective of the objective function, the approaches presented here lead to better solutions.

Chapter 4

Structure Quality Assessment

The most accurate assessments of a predicted structure can only be made after a prediction process (*e.g.* fragment assembly) is finished. For example, there are many properties that are informative but too expensive to compute repeatedly during prediction, such as pairwise atomic distances. Additionally, multiple different structures may be predicted for the same sequence using different prediction techniques, or a single technique may produce an ensemble of structures. Model quality assessment techniques select the best structure in these situations. Techniques for model quality assessment can be broadly divided into two categories: techniques that rely on properties internal to a given structure prediction, and techniques that rely on multiple predictions for a sequence.

The Critical Assessment of Structure Prediction (*CASP*) competition now includes a model quality assessment category [61], in which competitors are asked to submit a quality score for an entire structure, or to assign an error estimate to each residue. In CASP7, twenty-eight groups submitted full-structure quality estimates, and eight of those submitted per-residue error estimates. These eight groups assess structure quality using a variety of methods, which can be broadly organized into four categories.

4.1 Related Research

The first category of model quality assessment methods learns models to assess per-residue error based on intrinsic properties of the predicted structures. Techniques in

this category include ProQ [62], Victor/FRST [63] and QUAD [64]. ProQ uses neural networks to learn a model based on atom-atom contacts, residue-reside contacts, and agreement with predicted secondary structure. Victor/FRST uses a linear combination of four potential energy functions: a pairwise potential, a solvation potential, a torsion angle potential and a hydrogen bond potential. The relative weights for each function are optimized on the CASP4 decoy set. QUAD combines secondary structure, hydrogen bonding, and solvent accessibility information to produce a fitness score for each residue relative to its structural environment.

The second category forms a prediction based on the quality of a target-template alignment. Techniques in the second category include ProQProf [65] and SUBWAy [66]. ProQProf trains a neural network on profile-profile comparisons. For a target sequence, SUBWAy generates multiple suboptimal alignments to a template, then calculates the average deviation between the suboptimal alignments and the optimal alignment. A higher average deviation indicates a lower quality structure.

The third category determines the error by taking into account different per-residue error predictors. Techniques in the third category include ProQlocal [65] and Meta-MQAP [67]. A prediction by ProQlocal is a sum of the scores from ProQProf and ProQ. Meta-MQAP queries other quality assessment servers and combines the results.

The fourth category relies on the consensus of the predictions obtained from different protein structure prediction methods. Pcons [68] is alone in fourth category, and relies on the assumption that within an ensemble of structures predicted from different methods for the target protein, recurring structures and structural motifs have an increased probability of being high quality. Pcons determines the quality of a structure in two steps. First it performs pairwise LGscore alignments [69] between the query structure and all structures in the ensemble, then it uses the average S-scores [70] computed from these alignments to determine the per-residue error predictions. The results of CASP7 show that Pcons is the most successful approach in predicting both complete structure quality and per-residue errors, significantly outperforming the next best scheme.

4.2 Methods

Our algorithms produce per-residue error estimates using consensus based methods. We examine several static methods for consensus prediction, and determine that different predictors provide enough consistency for machine learning models to be effective. We present the use of a linear perceptron, standard regression, support vector regression and a simple weight learning technique to learn an appropriate aggregation scheme.

4.2.1 Dataset Construction

The data used in this study come from the CASP6 and CASP7 competitions. There are 66 target proteins and 57 predictors from CASP6, and 95 target proteins and 80 predictors from CASP7. Each of the CASP datasets can be viewed as a matrix, where the columns correspond to the different predictors, and the rows of the matrix are the sets of predictions for each residue of each protein. An (i, j) entry such a matrix D represents the distance between a residue of the row structure to the corresponding residue in the column structure. These distances are derived after superimposing the structures using LGA.[1] We will refer to the matrices for the CASP6 and CASP7 datasets individually as the CD6 and CD7 matrices, respectively.

4.2.2 Filling Missing Values

If a predictor in column j did not submit a structure for a protein A, the entries for all rows i in D corresponding to protein A will be empty. In addition, individual $D(i, j)$ entries can be missing because a predictor did not provide predictions for these residues. The scheme that we use to fill in these missing values is based on techniques developed in the field of collaborative filtering [72, 73, 74]. Let D be one of the CD6 or CD7 matrices, let $\mu_i = (\sum_j D(i, j))/nz_i)$ be the mean value of the nz_i non-zero entries of row i in D, let D' be the matrix obtained from D by subtracting from each non-empty value its row average (i.e, $D'(i, j) = D(i, j) - \mu_i$), and let $\mu_j = (\sum_i D'(i, j))/nz_j$ be the mean value of the nz_j non-zero entries of column j in D'. Each missing value at i, j is set to $\mu_i + \mu_j$. Note that in the case of filling missing values in a testing set the values

[1] We use the LGA program[71] to align each pair of structures with the same options used by the CASP assessors: `-3 -ie -d:4.0 -o0 -sda`

for μ_j from the training set are used.

Each entry in the matrix D has its own context, but subtracting μ_i from each row places all of the rows into a generalized context, and allows for the accurate computation of μ_j values. By subsequently adding μ_i to μ_j, the method provides an estimate of what the μ_j value should be in the context of the original row i. The advantage of this method is that it accounts for differences in the underlying structural alignments.

The advantage of filling missing entries with estimated values rather than zero is that missing values assigned to zero can confuse learning algorithms. If the original $d_j(a_i)$ value is just missing, then a zero is uninformative. However, if the zero represents a true $d_j(a_i)$ value of zero then the distance between A_S and A_S^k at residue i is zero. This means that the two structures align perfectly at this position.[2] In this case $d_j(a_i)$ should be treated as a zero, but in the former case $d_j(a_i)$ should simply be ignored.

To control for estimated values confusing machine learning algorithms, we create a custom training set for each test position encountered. For a given test position, the training set contains only those examples that have at least the same set of predictors present as those in the test position. As the set of positions tested by the custom training set technique are a subset of the CD6 and CD7 datasets, we refer to them as the PD6 and PD7 datasets, respectively. (The P stands for "partial".) PD6 contains 96,089 positions while PD7 contains 198,612 positions.

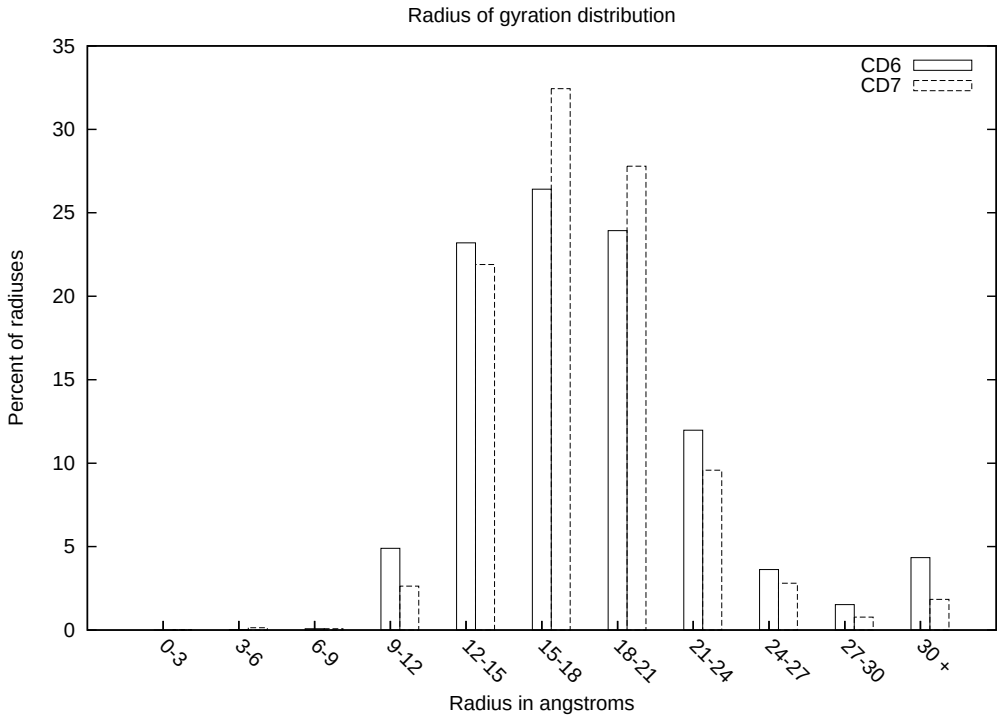
4.2.3 Dataset Composition

The CD6 matrix contains data for 824,145 positions and the CD7 matrix contains data for 918,467 positions. Figure 4.1 shows a histogram of the radius of gyration[3] for proteins in both the CD6 and CD7 datasets, while Figure 4.2 shows a histogram of sequence lengths.

As Figures 4.1 and 4.2 show, the proteins in our datasets have a range of sizes. The proteins in both sets are also structurally diverse, being spread across a variety of Structural Classification of Proteins (SCOP) folds, with 33 unique folds in the CD6 set, and 37 unique folds in the CD7 set. For the CD6 set, the minimum occupancy for a fold was 1, the maximum was 4, and the average was 1.3. For the CD7 set, the minimum

[2] We will use the terms *residue* and *position* interchangeably.
[3] The radius of gyration measures the structural size of a protein.

Figure 4.1: Radius of gyration distribution for both CASP datasets

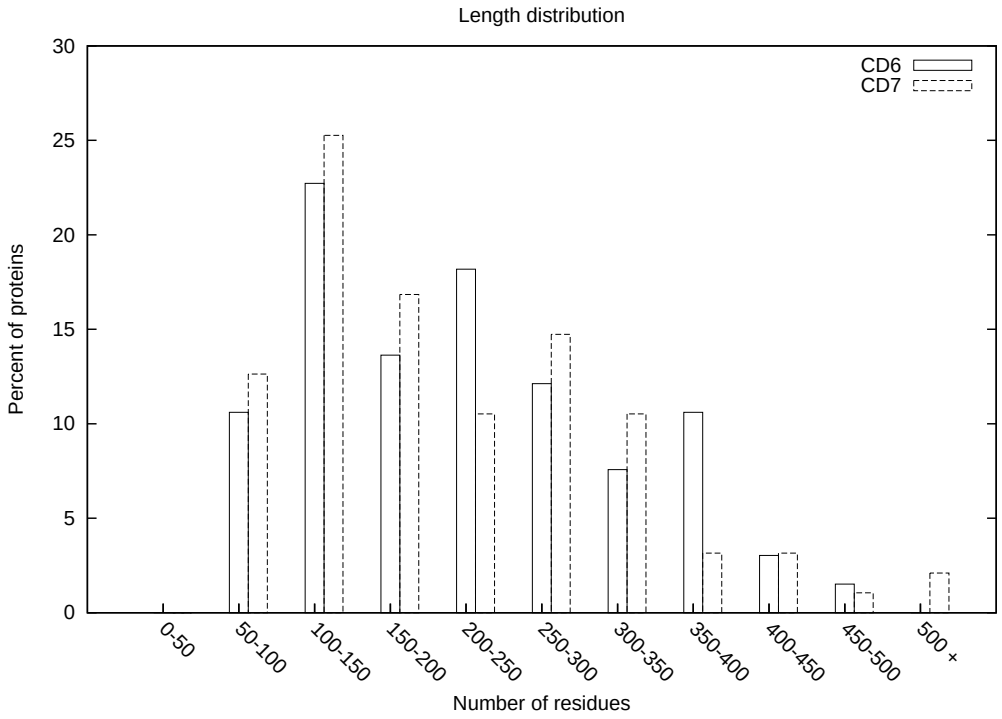


occupancy was 1, the maximum was 15, and the average was 2.1.

4.2.4 The Pcons Algorithm

We compare our methods with Pcons, the best performing model quality assessment algorithm in CASP7. Pcons values are taken from the CASP website with the exception of some incorrect values [61]. The results below reflect corrected values obtained from the Pcons authors. Pcons uses LGscore-based structural alignments in place of the LGA minimization employed by our techniques. Let p be a predictor, and let $LG_p(a_i)$ be the distance between the ith residue of A_S and the ith residue of A_S^p obtained after structurally superimposing A_S and A_S^p using the LGscore algorithm [69]. Pcons uses

Figure 4.2: Length distribution for both CASP datasets



the following three equations to produce a prediction, labeled $pc(a_i)$.

$$\tag{4.1}$$

$$S\text{-score}_p(a_i) = \frac{1}{1+(\frac{LG_p(a_i)}{\sqrt{5}})^2} \tag{4.2}$$

$$S\text{-score}(a_i) = \frac{1}{k}\sum_{p=1}^{k} S\text{-score}_p(a_i) \tag{4.3}$$

$$pc(a_i) = \sqrt{5}\sqrt{\frac{1}{S\text{-score}(a_i)} - 1} \tag{4.4}$$

Table 4.1: Prediction performance of the weight-learning methods with filled values.

	CD6		CD7	
Method	CC	RMSE	CC	RMSE
Support Vector Regression	0.69	10.21	0.80	6.36
Linear Perceptron	0.68	11.31	0.79	7.45
Standard Regression	0.69	**9.48**	0.80	**6.26**
Constrained Regression	**0.71**	10.04	**0.81**	6.37

Values in this table represent the Pearson correlation coefficient (CC) and root mean squared error (RMSE) between the predicted and true per-residue distances. Boldfaced entries correspond to the best performing scheme for each dataset and performance assessment metric.

4.2.5 Static Consensus Error Prediction Methods

Our consensus-based per-residue error prediction method is rooted in the same principles introduced by Pcons. The general procedure works as follows. Let A be the amino acid sequence of the query protein and let A_S be its predicted 3D structure. Let A_S^t be the true 3D structure for A and let $\{A_S^1, A_S^2, \ldots, A_S^k\}$ be the structures of A predicted by k different structure prediction methods. For each residue a_i of A, let $d_k(a_i)$ be the distance between the ith residue of A_S and the ith residue of A_S^k obtained after structurally superimposing A_S and A_S^k using the LGA program [71]. The predicted distance $d(a_i)$ between residue a_i in A_S and A_S^t (the error estimate for position i) is given by

$$d(a_i) = \sum_{p=1}^{k} w_p d_p(a_i), \tag{4.5}$$

where w_p is a weight associated with the pth predictor and $\sum_p w_p = 1.0$. The idea behind this approach is that each of the k structures can be treated as an expert's prediction for A's real structure. Thus, the per-residue error can be determined by a weighted average over the per-residue distances between A_S and the A_S^p structures for each predictor p. The various w_p parameters control how the distances between the query and the predictions are weighted. A straightforward approach is to treat each of the predictors equally by setting these weights to $1/k$, which is the same as averaging the predictions.

The above method differs from Pcons in two important ways. First, it uses LGA

alignments as opposed to LGscore alignments. An LGA alignment is constructed by attempting to optimize both the longest continuous sequence that can fit under a certain cutoff, as well as a global measure of alignment quality. An LGscore alignment attempts to find the most significant non-continuous segment of a model [70]. Second, our method averages raw distances as opposed to S-scores (Equation 4.1).

4.2.6 Learning-based Consensus Error Prediction Methods

Machine learning techniques have been specifically designed for optimizing weights like those in Equation 4.5, so we developed techniques that, rather than treating each of the predictors equally, use these methods to estimate the weights directly from the data. We formulate learning the weights as the following supervised learning problem. Given a set of training examples a_i, each described by the tuple $(d_t(a_i), \langle d_1(a_i), d_2(a_i), \ldots, d_k(a_i)\rangle)$, where $d_t(a_i)$ is the actual distance between the ith residue of A_S and the ith residue of A_S^t in an alignment between $A_S and A_S^t$, learn the set of w_p values (see Equation 4.5) that minimize

$$\sum_{a_i} \left(d_t(a_i) - d(a_i)\right)^2 . \tag{4.6}$$

4.2.7 Evaluating Weight Learning Algorithms

Using the CASP6 and CASP7 datasets, we evaluate three training approaches for learning the weights described in Section 4.2.5: unfilled, filled, and custom. The unfilled and filled approaches are evaluated using a leave-one-*protein*-out framework. A protein is selected, and all positions from this protein are assigned to the test set. All remaining examples are used as the training set. The whole process is repeated for each available protein, so every position in the CD6 and CD7 matrices is used for testing at some point. The difference between the unfilled and the filled approaches is that in the latter, the empty entries of the CD6 and CD7 matrices are filled using the method described in Section 4.2.2.

The custom training set approach is evaluated under a leave-one-*position*-out framework. A single position a_i serves as the test set and all positions from proteins other than A are searched to find the training set. The training set for position a_i corresponds to those rows of the CD6 and CD7 matrices whose non-empty columns are a super-set

of the non-empty columns of the row for a_i. This creates a training set with no missing values: the sub-matrix formed by the rows corresponding to the training set and the set of columns in the test position are completely filled. It might seem that this approach requires learning 824,145 models for CASP6 and 918,467 models for CASP7 (one for each row of the CD6 and CD7 matrices). However, predictions are usually made for every residue in a protein, or none at all. This means that the non-empty columns for all the rows of the same protein will usually be the same (leading to the same training sets), so the number of required models is actually much smaller. To further reduce the number of models, each model that we learn has to be used to test at least ten positions. This reduces the total number of models learned to only 553 for CASP6 and 1064 for CASP7.

4.2.8 Weight Learning Algorithms

Linear Perceptron

Algorithm 1 Learning Weight Vectors with the linear perceptron algorithm

Input: k: Number of Predictors.
nts: Number of Training Samples.
N: Number of Training Iterations.

Output: w: Weight Vector.

$w \leftarrow 1/k$
for $n = 1$ to N **do**
  for $x = 1$ to nts **do**
    $e_p \leftarrow |d_p(a) - d_t(a)| \ \forall_p$
    $\phi_j \leftarrow 1/(e_p + \sum_p e_p/k) \ \forall_j$
    $\phi \leftarrow \phi/\|\phi\|_1$
    $\alpha \leftarrow |d(a) - d_t(a)|/m$
    $w \leftarrow w + \alpha\phi$
    $w \leftarrow w/\|w\|_1$
  end for
end for
Return w

One way of solving the learning problem described in Section 4.2.6 is to use Rosenblatt's linear perceptron classifier [75]. This is an online learning algorithm that iteratively updates a weight vector w for each training example a based on the difference between its actual and predicted values. Pseudo-code for our implementation of this algorithm is shown in Algorithm 1. For each position, the linear perceptron determines the error of each predictor (line 4). Each predictor is then assigned a weight that is inversely related to its error and the vector of these weights ϕ is scaled to sum to one (lines 5 and 6). Note that in line 5, the $\sum_p e_p/k$ factor is used to reduce the difference between lower and higher weights, as we find that using this smoothing factor improves results. The learning rate α is updated based on the error of the prediction for each example $d(a)$, as determined using Equation 4.5. In the case of the unfilled CD6 and CD7 datasets, we use $d(a)/\sum_p w_p$ as the prediction. This prevents the sparsity of the set from skewing the values learned for w. The vector ϕ becomes the update to w (line 8), and w is scaled to sum to one after processing each training example (line 9). The final weights are the values of w after five iterations over the training data, as a preliminary test (results not shown) showed a small difference between the weights from the fourth and fifth iterations. Note that this is a variation on a traditional linear perceptron, in which w is updated according to $w \leftarrow w + \alpha(real - predicted)d$. We use the variation shown in Algorithm 1 because it performs better for our problem.

Support Vector Regression

We use the SVMLight implementation for support vector regression [76]. Default values are used for the tube width and regularization parameters. The details of this regression technique have been described in detail elsewhere [77] and will not be covered.

Standard and Constrained Regression

We use Matlab for standard regression and for constrained regression in which the weights w_p must be positive. We also experimented with a second constrained regression formulation, in which the weights w_p of the predictors must be positive and sum to one. This regression formulation could not learn an appropriate set of weights in the majority of cases, so the results are not included.

Table 4.2: Prediction performance of the static methods.

	CD6		CD7	
Method	CC	RMSE	CC	RMSE
LGA–Distance	0.68	11.71	**0.79**	**7.49**
LGA–*S*-score	0.66	**11.06**	0.74	7.78
LGscore–Distance	0.66	11.46	0.77	7.66
LGscore–*S*-score	**0.70**	11.19	0.76	8.18

Values in this table represent the Pearson correlation coefficient (CC) and root mean squared error (RMSE) between the predicted and true per-residue distances. Boldfaced entries correspond to the best performing scheme for each dataset and performance assessment metric.

4.3 Results

We examine the performance obtained by four static approaches, and four machine learning approaches.

4.3.1 Static Approaches

Two of the static schemes (LGA–Distance and LGA–*S*-score) use LGA to compute alignments, while the other two schemes (LGscore–Distance and LGscore–*S*-score) use LGscore to compute alignments. All four use average raw distances or *S*-scores. Using LGscore alignments and averaging *S*-scores is identical to the Pcons approach. The prediction performance achieved by these methods on CD6 and CD7 is shown in Table 4.2. The performance is assessed by calculating both the Pearson correlation coefficient (CC) and the root mean squared error (RMSE) between the actual and predicted per-residue errors.

These results show that the performance of the various schemes differs between the two datasets. For CD6, LGscore–*S*-score achieves the best correlation while LGA–*S*-score obtains the lowest RMSE. For the CD7 dataset, LGA–Distance shows the best performance both in terms of correlation and RMSE. The improvements achieved by the best performing schemes when compared to the next best schemes are significant at $p \leq 0.0001$. Comparing the performance of the schemes using distance-based averaging over those that average *S*-scores, we see that the former lead to better results on CD7,

Table 4.3: Prediction performance of the weight-learning methods.

	CD6		CD7	
Method	CC	RMSE	CC	RMSE
Support Vector Regression	0.68	10.35	0.80	6.41
Linear Perceptron	0.69	9.74	0.80	6.46
Standard Regression	**0.70**	9.32	0.80	6.28
Constrained Regression	**0.70**	**9.20**	**0.81**	**6.19**

Values in this table represent the Pearson correlation coefficient (CC) and root mean squared error (RMSE) between the predicted and true per-residue distances. Boldfaced entries correspond to the best performing scheme for each dataset and performance assessment metric.

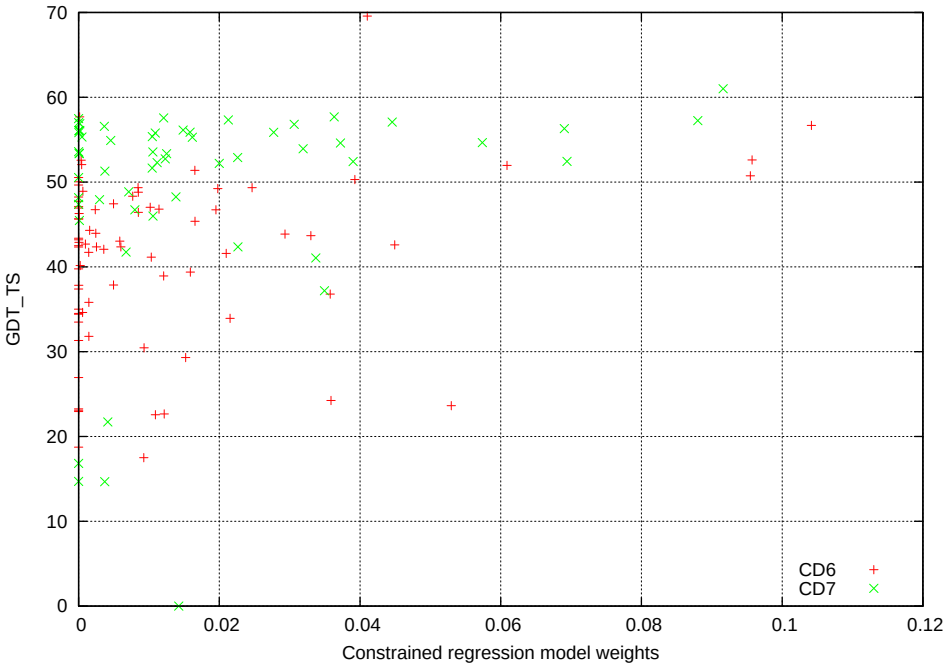
and mixed performance on CD6. A similar dataset specific behavior is observed when comparing LGA vs LGscore. LGA-based alignments perform consistently better on CD7 whereas the relative performance of LGA and LGscore-based alignments on CD6 changes based on the averaging scheme (distances or S-scores) and performance metric.

4.3.2 Machine Learning Approaches

Most of the learning schemes (Table 4.3) outperform the static prediction methods (Table 4.2) on both the CD6 and CD7 datasets. The overall best results for both datasets are obtained by the constrained regression method, which shows improved RMSE values of 20.2% and 21.0% over the best performing static schemes for CD6 (LGA–S-score) and CD7 (LGA–Distance), respectively. Constrained regression performance in terms of correlation is similar to that of the best static scheme for CD6 and better by 3.4% for CD7. Note that the relatively higher gains in terms of RMSE when compared to the improvements achieved in terms of correlation result from the objective function of the learning methods (Equation 4.6) directly minimizing the RMSE of the predictions. Overall, the prediction improvements and consistency achieved by the four learning schemes (and constrained regression in in particular) show that learning methods can learn weights that are better than simple averaging.

Ideally, the learned weights should reflect the quality of the prediction servers. Figure 4.3 plots the weights learned by the constrained regression method against the quality of the structures generated by each server. The quality of a predicted structure

Figure 4.3: Model weights vs server quality



is measured by its global distance test total score, or *GDT_TS*, produced by the LGA program [71]. Values for the model weights and server GDT_TS scores correspond to the averages over the 80 and 58 structures used in the CD6 and CD7 datasets, respectively. From these plots we see that even though there is a correlation between the weights and the GDT_TS scores (0.24 for CD6 and 0.22 for CD7), this correlation is far from perfect. There are high quality servers that are assigned low weights as part of the training, indicating that the information provided by them is redundant for estimating the per residue error. In contrast, there are servers that do not perform extremely well, but they are still utilized by the learned model, suggesting that they provide key information for improving quality assessment.

Eliminating Missing Predictions

Table 4.4 shows the results from testing models built using custom data sets. As these results are obtained on the PD6 and PD7 datasets they are not directly comparable to those reported in Tables 4.3–4.1, which were obtained on the CD6 and CD7 datasets.

Table 4.4: Performance of various interaction prediction methods for the PD6 and PD7 datasets.

	PD6		PD7	
Method	CC	RMSE	CC	RMSE
Custom training				
Support Vector Regression	**0.87**	**3.33**	**0.89**	**2.86**
Linear Perceptron	**0.87**	3.81	**0.89**	2.93
Standard Regression	0.70	6.02	0.82	3.80
Constrained Regression	**0.87**	3.46	**0.89**	**2.86**
Global training				
Support Vector Regression	**0.90**	**2.97**	**0.90**	**2.67**
Linear Perceptron	0.88	3.64	0.89	2.90
Standard Regression	0.89	3.20	0.90	2.76
Constrained Regression	0.89	3.15	0.89	2.79
Static consensus				
LGA–Distance	**0.86**	4.29	**0.89**	**3.19**
LGA–*S*-score	**0.86**	**3.74**	0.83	3.56
LGscore–Distance	0.67	5.56	0.72	4.48
LGscore–*S*-score	0.81	4.57	0.79	4.33

Values in this table represent the Pearson correlation coefficient (CC) and root mean squared error (RMSE) between the predicted and true per-residue distances. Boldfaced entries correspond to the best performing scheme for each dataset and performance assessment metric.

For this reason, Table 4.4 also shows the performance of the weight-learning methods trained on the entire training set and the static consensus error prediction methods. These results are shown under the headings "Global training" and "Static consensus", respectively. Analyzing the performance of the custom training methods, constrained regression once again shows consistently good performance, with the best correlation coefficient for both PD6 and PD7. Constrained regression also shows the best RMSE for PD7, and is within 4% of the best RMSE (3.46 as compared to support vector regression at 3.33) for PD6. However, comparing the results obtained by the custom training methods to those obtained by the methods trained using the entire training set, we see that using the entire set achieves consistently better performance, even when there are missing values. The size of the training set of each custom training problem is relatively small, and as such the models are not as effective as those learned on the entire dataset.

4.4 Discussion

The results presented in this study reveal several interesting trends. First, a machine learning framework can be utilized to learn models that intelligently weight different structures in the context of consensus-based error prediction. Second, constrained regression outperforms or performs on par with more complicated learning techniques while preventing over-fitting. This performance is consistent across two CASP datasets. Third, filling missing values with an approximation does not produce consistent gains in performance. Fourth, customizing training sets to assist machine learning techniques does not produce substantial gains in performance over static prediction methods. Moreover, machine learning methods trained on these sets perform considerably worse than models trained using all available training data, even when these data contain missing values.

All of the above lays the groundwork for the construction of a regression model trained on data from existing prediction methods. The models described above learn how to weight the predictions of a fixed set of protein structure prediction methods. To use these models for assessing the quality of a protein's predicted structure, the protein needs to be predicted by the same set of prediction methods on which the model was trained. In our study, training and testing of these models is done in the context of a CASP competition, but models of this type can also be trained and applied outside of CASP. Specifically, weights for a set of structure prediction servers will be learned by the machine-learning methods. By querying this set of servers with sequences from recently released structures, the resulting predictions can be used for training. If a server alters its prediction algorithm the models may not work properly, but as constrained regression does not require much time to train, the models could be continually updated. Moreover, as more advanced prediction servers become available, the underlying models can be retrained to use the new set of prediction methods. Note that restricting the training set to recently characterized structures helps ensure that the prediction servers actually make predictions, rather than simply querying a database for the true structure. Finally, as the results in Figure 4.3 indicate, for many servers, the weights that are learned are either zero or very small, suggesting that the above framework can achieve good performance with a relatively small number of servers.

Chapter 5

Interacting Residue Prediction

Being able to disrupt or enhance protein-protein interactions has extensive applications in the pharmaceutical industry and in the development of experimental methods for studying and understanding biological systems. A key step towards this goal is the identification of a protein's residues that bind to other proteins in order to form protein complexes. While Chapter 2 focuses on producing better protein sequence alignments, in this chapter we rely on high quality alignments to predict which residues of a protein are involved in interactions.

The development of high-throughput experimental techniques for identifying protein interactions has enabled large-scale studies of protein-protein interaction networks [78, 79, 80, 81, 82, 83]. However, these experimental techniques do not identify which protein residues are involved in interactions. Instead, interaction information is primarily provided from X-ray crystallography of protein complexes, which is both expensive and time consuming.

5.1 Related Research

In recent years, several reviews have been published that recount the breadth of interaction prediction techniques [84, 85, 86]. These techniques can be broadly divided into two groups. The first group makes predictions based on structural information, while the second group uses only sequence information. Current databases contain millions of sequences, but only tens of thousands of structures. As this gap continues to widen,

accurate computational methods that predict interacting residues from sequence alone are increasingly in demand.

Prediction methods based on structure typically use properties such as surface shape, sequence conservation [87, 88], electrostatic information, hydrophobicity [87], residue interface propensity [89] and solvent accessible surface area [90, 87, 91, 92, 89]. In the absence of structural information, predicted structural properties have also been used [93]. Combinations of these properties are typically fed into machine-learning methods to build models that can predict which residues interact with other proteins. (These are also referred to as *binding* residues). The most common machine-learning methods are based on neural networks [87, 92, 89] and support vector machines [94, 95, 91].

5.2 Methods

Most existing models for sequence-based prediction rely on residue-local information, but our method, the Sequence-based Interacting Residue Predictor (SIRP), learns a prediction model that combines both locally and globally derived features for each residue. The local features are based on sequence profiles computed using PSI-BLAST [7] and predicted secondary structure elements. The global features are homology-based transfer scores (see [96] and Section 5.2.5), derived from alignments against proteins with known protein binding residues. A homology-based transfer score for a given residue encodes global structural constraints implicit in these alignments.

Evaluating methods that predict protein binding residues is challenging due to a lack of standard datasets and a lack of agreement on what constitutes an interaction [84, 85, 86]. To quantify what constitutes an interacting residue, some studies use a distance-based cutoff, some a loss of solvent accessible surface area, and some use both (see [84] for an excellent compilation of the various definitions in use). While an accurate description of an interface is important, the lack of consensus makes it difficult to compare prediction methods.

To help ensure that the assessment of our methods is accurate, we use a larger dataset than has previously been reported in protein-protein interaction studies. This dataset is derived from the PiSite interaction database [97], and includes over 4,000 sequences

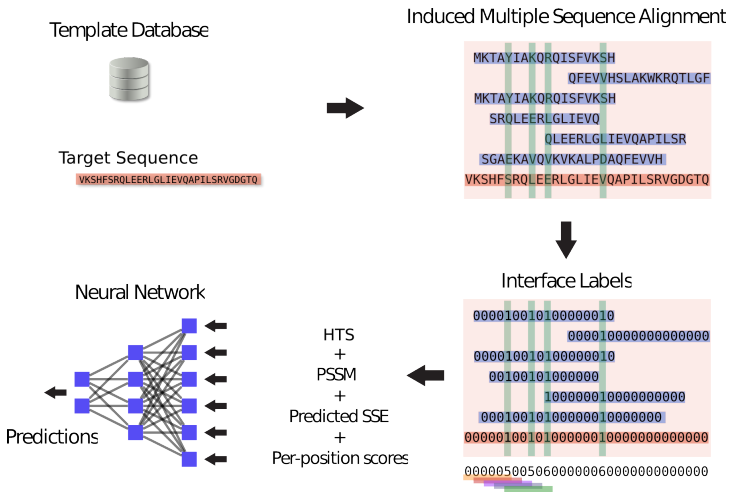


Figure 5.1: Overview of the SIRP technique.

with a pairwise sequence identity less than 25%. Using these sequences, we compare an existing approach called ISIS [93] with several variations of SIRP. SIRP outperforms ISIS, achieving areas under the receiver-operating characteristic and precision-recall curves (*ROC* and *PR*) of 0.656 and 0.326, compared with 0.615 and 0.270 for ISIS, respectively. In addition, we develop a simple approach to determine, *a priori*, the set of proteins for which SIRP is expected to perform well. When evaluated on this set SIRP achieves an ROC of 0.768 and a PR of 0.445, which is substantially higher than ISIS, at 0.640 and 0.290, respectively.

5.2.1 SIRP

An overview of the SIRP technique is shown in Figure 5.1. Given a protein whose binding residues are not known, referred to as the *target* protein, SIRP starts by searching a database of protein sequences with known protein binding residues, referred to as *template* proteins, for high-scoring alignments to the target. These alignments are assembled into an induced multiple sequence alignment. Scores are assigned to each residue of the target based on the number of aligned protein-binding template residues.

(See section Section 5.2.5 for a description of how these scores are calculated.) Information in a window (represented by overlapping colored bars in Figure 5.1) around each residue of the target protein is used to construct a feature vector. For each position in the window, components of this feature vector consist of the homology-based transfer score, the profile-based alignment score for this position, the row from the position-specific scoring matrix (PSSM), and the predicted secondary structure vector. These features are then used as input to a neural network model.

5.2.2 Alignment Scoring Matrix Construction

As in Chapter 2, evolutionary profiles are generated using PSI-BLAST [7] and version 2.2.12 of the NCBI NR database, which contains 2.87 million sequences. Both the position specific scoring matrix ($PSSM$) and position specific frequency matrix ($PSFM$) generated by PSI-BLAST are employed by our prediction and alignment methods. Five iterations are used in PSI-BLAST with an e-value threshold of 0.01 for inclusion in the profile using the command line options `-j 5 -e 0.01`.

For both template and target proteins, we obtain secondary structure predictions using YASSPP [98]. Each row in a secondary structure element matrix corresponds to a residue in the protein, with the columns corresponding to predictions for three secondary structure states: helix, coil and sheet. A bigger value in this matrix corresponds to a greater likelihood that a given residue will adopt a given secondary structure.

5.2.3 Alignment Scoring

The alignment scoring scheme that we use is derived from Picasso [99], a high quality profile scoring function [100, 101]. We align sequences by computing an optimal alignment using an affine gap model which scores aligned residues i and j in sequences X and Y, respectively, using a combination of profile-profile scoring and secondary structure matching. The score is stored in row i and column j of a matrix m, where

$$
\begin{aligned}
m_{i,j} &= \sum_{k=1}^{20} PSSM_X(i,k) \times PSFM_Y(j,k) \\
&+ \sum_{k=1}^{20} PSSM_Y(j,k) \times PSFM_X(i,k) \\
&+ w_{SSE} \sum_{k=1}^{3} SSE_X(i,k) \times SSE_Y(j,k),
\end{aligned} \tag{5.1}
$$

$PSSM$ and $PSFM$ are generated by PSI-BLAST, and SSE is a position-specific matrix encoding the secondary structure associated with each position. For a sequence with n residues, the sizes of the matrices are $n \times 20$, $n \times 20$, and $n \times 3$, respectively. The parameter w_{SSE} is the relative weight of the secondary structure score, set to $w_{SSE} = 3$ based on our experience in [96].

5.2.4 Alignment Parameters

We compared the local, global and global end-space free alignment techniques when aligning target and template proteins. (See Chapter 2 and [32] for more information on these alignment algorithms). The variations in performance between the three were slight, with the local technique edging out the other two, so we use local alignments in our prediction methods.

Three parameters influence the quality of an optimal local alignment using a scoring matrix m: gap-opening cost (go), gap-extension cost (ge), and the zero-shift value (zs). The gap-opening cost is incurred once per series of spaces in an alignment, and the gap-extension cost is incurred for each space inserted after the first. See Chapter 2 for a detailed explanation of go, ge and m as their behavior is the same between local and global alignments. The zero-shift value is added to m in an attempt to ensure that it has an average value of zero. This is necessary for meaningful local alignments, as the alignment algorithm will only stop extending a putative alignment when the score dips below zero.

As we are using local alignments, we cannot use parameter values from Chapter 2, as they are optimized for global alignments. In order to determine the best values for go, ge and zs under a local alignment, a small study was performed on a set of 15 proteins.

These proteins are not included in the reported results. We optimize these parameters using a grid search with *go* and *ge* ranging from 0.5 to 5 and *zs* ranging from -5 to 5, in increments of 0.5. In addition, we test scaling each matrix m individually for each pair of proteins so that the average value is zero. We find that the parameters providing the best performance are a gap opening cost of 1.5, a gap extension cost of 1.0 and a zero-shift of 1.5. Results for SIRP below are based on using these values.

5.2.5 Homology Transfer Score

The goal of SIRP is to predict whether a residue in a target protein is involved in a protein-protein interaction, and a key part of this method is the homology transfer score, or *HTS*. An HTS can be thought of as a vote among the k best alignments with a target, where high quality template matches influence the vote more than poor matches. To accomplish this, we weight template residues by the score of a partial alignment between target and template in a window around the aligned target residue. This means that every residue in a given partial alignment receives the same weight. We explore window sizes of $w \in \{1, 2, 3, 4, 5\}$ and find that using $w = 4$ results in the best performance. Each w-size window involves $2w + 1$ residues, centered on the residue under consideration. If the smallest weight on a template residue associated with a target residue is negative, the weights of template residues are shifted up so that the lowest weight is equal to one. For each target residue, the associated template residues also have their weights normalized so that they sum to one. Target sequence interaction predictions are made by summing a subset of weights of the aligned positions in the templates. Interacting template residues add their weight to this sum while non-interacting residues and gaps add nothing. This results in a prediction score between zero and one for each residue.

5.2.6 Feature vectors

The final stage of SIRP involves training a neural network model to differentiate between interacting and non-interacting residues. The input to the neural network consists of feature vectors representing single residues. For each of these, a window of size w is used to construct a vector containing four pieces of information for each residue. These are: (i) its HTS, (ii) its average Picasso-based alignment score against the other sequences in

the multiple sequence alignment used to derive the HTS, (iii) its row from the PSSM, and (iv) its YASSPP-predicted secondary structure vector. Thus, the total set of features describing each residue is $1 + 1 + 20 + 3 = 25$, leading to an input vector consisting of $25 \times (2w + 1)$ features.

Each feature type provides a distinct piece of information. The HTS provides a surrogate for the probability that a residue is involved in protein binding. The alignment scores indicate the quality of the HTS, as poorly aligned sequences result in inaccurate surrogates. The row of the PSSM provides information about the sequence conservation around the position being predicted, while the secondary structure predictions provide a notion of local structure.

5.2.7 Identification of Reliable Predictions

By restricting SIRP to alignments expected to be of high quality we produce more reliable predictions. To this end, we explore three measures for estimating the quality of a alignment. These are the fraction of the target aligned, F_g (g for tar*g*et), the fraction of the template aligned F_p (p for tem*p*late), and the average per-position score of an alignment P_s. For a given alignment between a target and a template, the first two are calculated as

$$F_g = \frac{\text{number of aligned residues}}{\text{length(target)}} \tag{5.2}$$

and

$$F_p = \frac{\text{number of aligned residues}}{\text{length(template)}} \tag{5.3}$$

where length(target) is the total number of residues in the target, and length(template) is the total number of residues in the template.

The average per-position score of an alignment between X and Y is calculated as

$$P_s = \frac{\sum_{i,j} m_{i,j}}{\text{number of aligned residues}} \tag{5.4}$$

where $m_{i,j}$ is defined as in Equation 5.1, and residues aligned with gaps are ignored. While F_g and F_p will by definition fall between zero and one, P_s will depend on the length of the alignment. To allow for an alignment-independent threshold on this value, we normalize P_s as follows. Let GP be the average per-position score of the target aligned against the template. Let GG be the average per-position score of the target aligned

against itself. Let PP be the average per-position score of the template aligned against itself. Then the normalized per-position score (P_s^n) for aligning a target-template alignment is

$$P_s^n = \frac{2GP}{GG + PP}. \tag{5.5}$$

P_s^n serves as the first measure used for a cutoff threshold. The second measure used as a threshold is F_p, as in preliminary experiments using a threshold for either F_g or F_p is sufficient to improve performance. Hence, given a set of alignments to template proteins for a given target, we use the top 20 scoring alignments (see Equation 5.1) with F_p and P_s^n above 0.8. If there are fewer than 20 alignments that meet these criteria, fewer than 20 are used.

5.2.8 Dataset

Our dataset is derived from the PiSite database of protein interaction sites [97]. Key to the construction of this database is the consideration of multiple interacting partners for a given protein chain, because in order to accurately characterize a protein-protein interface, all known interactions must be taken into account. In PiSite, pairs of proteins are considered interacting if they have at least two atoms within 4Å of each other. The dataset is constructed using biological units, rather than asymmetric units, in order to eliminate crystallographic interfaces. The solid design and broad coverage of this dataset makes it a good candidate for an interaction prediction benchmark.

Starting from the non-redundant set of proteins distributed by PiSite, we exclude any proteins without interacting residues. This set is then filtered at 25% identity using cd-hit [102], leaving 4183 protein chains. Of these, 15 are reserved for alignment parameter tuning, leaving 4168 proteins containing 714208 residues, of which 19.77% are labeled interacting, and 80.23% are not. We will refer to this dataset as DS1. Proteins are randomly split into ten separate folds for cross-validation. Note that because the entire set is filtered at 25% identity, no protein in a test set has more than 25% identity with one in a training set. All of the methods reported here are trained and tested on the same data, allowing for a direct comparison between them.

Restricted Dataset

We leverage the confidence estimates described in Section 5.2.7 to improve the performance of SIRP. Because the quality of an alignment can be measured independently of the interface labels, the reliability of a given prediction can be accurately estimated. As such, we can choose not to make predictions which we know will be less reliable. We construct a second dataset called DS2, containing 828 proteins for which SIRP is expected to make confident predictions. These proteins are selected from DS1 based on the F_p and P_s^n values of the alignments with templates. If a query protein has at least three alignments to templates with $F_p > 0.8$ and $P_s^n > 0.8$ the query is selected for DS2. Cross-validation folds are the same for DS2 as for DS1, and the training sets used for DS2 are the same. Only the proteins used for testing differ between DS1 and DS2.

5.2.9 ISIS Predictions

The ISIS algorithm utilizes secondary structure predictions and predicted solvent accessibility information in addition to PSSMs [93]. The PSSMs used as input to ISIS are identical to those used in the SIRP method (see Section 5.2.3), but the secondary structure and solvent accessibility predictions are made by ProfPHD [103]. These data are used to construct feature vectors (see Section 5.2.6) which serve as input to a neural network. Note that we do not implement the post prediction refinement filter described in [93] as in our experiments it did not improve performance.

5.2.10 Neural Network Structure and Training

We use three-layer feed-forward neural networks, implemented by the libfann library.[1] The connection rate is set to one and the learning rate is set to 0.05, both of which are the defaults for libfann. Designing the architecture of a neural network is application specific, so to ensure that both the network employed by SIRP and the network employed by ISIS perform optimally, in a preliminary study we tested different values for both the number of hidden nodes ($hn \in \{50, 75, 100, 200\}$) and the number of training epochs ($te \in \{100, 200\}$). Somewhat surprisingly, our results showed little variation between

[1] http://fann.sourceforge.net

combinations of values for the different parameters. We report results for ISIS using $te = 200$ and $hn = 50$, and for SIRP using $te = 200$ and $hn = 100$, as these are the best performing settings for the respective schemes. Note that besides neural networks, we also performed a set of experiments using support vector machines as the underlying machine-learning approach, but saw no improvement.

5.2.11 Evaluation Metrics

We evaluate the performance of the different methods using two metrics: the area under the *ROC* curve [104], and the area under the precision-recall curve. The ROC curve is obtained by varying the threshold at which residues are considered interacting according to a value provided by the predictor. In the case of the neural network predictions, two values are produced: the negative output, representing the confidence that the input is a negative example, and the positive output, representing the confidence that the input is a positive example. The larger of the two values is taken as the prediction. Repeatedly setting a threshold value on these predictions and analyzing the resulting performance is used to produce the *ROC* curve. The area under this curve, referred to simply as *ROC*, summarizes the predictor behavior: a random predictor has $ROC = 0.5$ while a perfect predictor has $ROC = 1.0$. In general a larger *ROC* indicates better predictive power.

For any binary predictor, the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) determines standard classification statistics. These are

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{5.6}$$

$$\text{Recall} = \text{Sensitivity} = \frac{TP}{TP + FN}. \tag{5.7}$$

As [105] notes in a study of functional residue predictions, analyzing only an ROC curve can be misleading. As an alternative, they present precision-sensitivity plots, which we refer to as PR curves, as a means to compare performance. We provide this measure as well, summarized by the area under the PR curve (PR).

Table 5.1: Cross-validation results on DS1

Method	Overall		Per Protein			
	ROC	PR	μ_{ROC}	σ_{ROC}	μ_{PR}	σ_{PR}
ISIS	0.615	0.270	0.593	0.111	0.319	0.202
NN-PSSM+SSE	0.611	0.266	0.592	0.114	0.323	0.200
HTS	0.626	0.312	0.587	0.151	0.343	0.224
NN-HTS	0.635	0.324	0.596	0.167	0.355	0.232
NN-HTS+PPS	0.637	0.324	0.597	0.166	0.355	0.230
NN-HTS+PPS+ PSSM+SSE (SIRP)	0.656	0.326	0.614	0.145	0.310	0.202

Performance values based on 10-fold cross validation of the dataset. Values under "Overall" are the ROCs from all predictions across the 10 folds, while those under "Per Protein" are divided based on target protein.

Table 5.2: Cross-validation results on DS2

Method	Overall		Per Protein			
	ROC	PR	μ_{ROC}	σ_{ROC}	μ_{PR}	σ_{PR}
ISIS	0.640	0.291	0.623	0.104	0.307	0.166
NN-PSSM+SSE	0.641	0.290	0.629	0.104	0.310	0.168
HTS	0.765	0.500	0.748	0.149	0.486	0.244
NN-HTS	0.775	0.508	0.760	0.161	0.502	0.247
NN-HTS+PPS	0.777	0.493	0.761	0.158	0.498	0.244
NN-HTS+PPS+ PSSM+SSE (SIRP)	0.768	0.445	0.741	0.136	0.315	0.191

Performance values on the subset of proteins with F_p and $P_s^n > 0.8$ (see Section 5.2.7)

5.3 Results

The performance of six prediction methods is shown in Table 5.1. Results for our implementation of ISIS are listed at the top of the table. The next method down (NN-PSSM+SSE) differs from ISIS in that (i) it does not use predicted solvent accessibility and (ii) it uses YASSPP for secondary structure predictions. The third method (HTS) does not rely on neural networks, using the homology-based transfer scores as predictions directly. The NN-HTS method makes predictions using a neural network that takes the HTS values as input. The NN-HTS+PPS method is a neural network whose features consist of the HTS values along with the average per-position scores (PPS) taken from the induced multiple sequence alignment used in generating the HTS values. Finally,

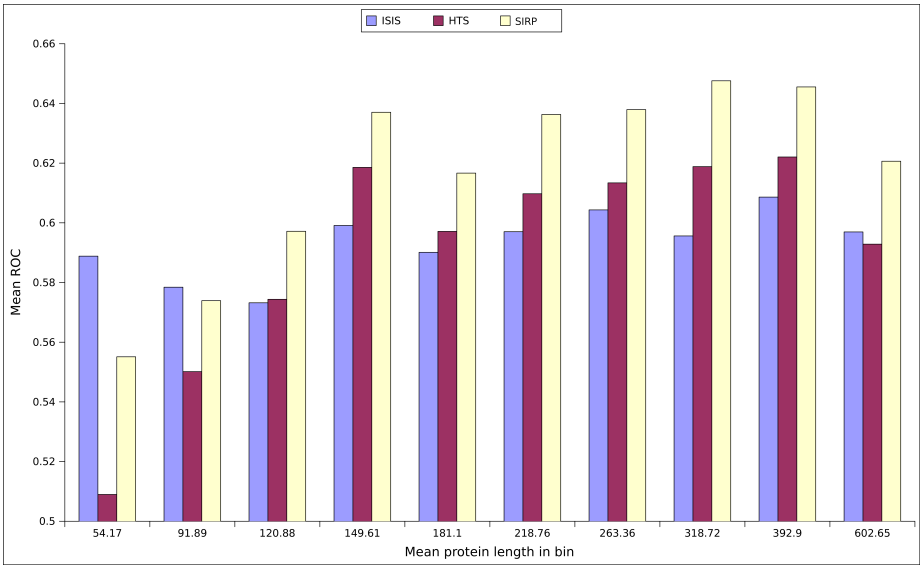


Figure 5.2: Per-protein ROC values binned on protein sequence length

NN-HTS+PPS+PSSM+SSE is the method in which the neural network combines all four types of features discussed in Section 5.2.6 (the SIRP method).

Examining the results in this table we see that for the overall ROC and PR scores, SIRP performs the best, whereas ISIS and NN-PSSM+SSE perform the worst. The overall ROC and PR scores achieved by SIRP are 0.656 and 0.326, respectively, whereas the corresponding numbers for ISIS are 0.615 and 0.270. SIRP also outperforms ISIS and the other methods in terms of the average per-protein ROC, but not PR. Also, the relative gains are not as high as those for the overall ROC. Generally, the performance achieved by the other methods that utilize HTS information is better than the methods that do not (i.e. ISIS and NN-PSSM+SSE), both in terms of overall ROC and PR scores. However, their performance gains in terms of the average per-protein ROC and PR scores are less consistent, even though NN-HTS and NN-HTS+PPS outperform both ISIS and NN-PSSM+SSE. A common characteristic with all the methods that utilize HTS information is that they have higher variability in terms of their per-protein predictions. This is analyzed further in Section 5.3.1. One interesting observation is that the HTS method, which relies only on the homology-based transfer scores, performs quite

well and achieves higher overall ROC and PR scores than ISIS and NN-PSSM+SSE. Finally, the performance of ISIS and NN-PSSM+SSE is very similar. This should not be surprising as these two methods utilize similar feature sets.

Table 5.2 shows the performance achieved by the various methods on the DS2 dataset. Recall from Section 5.2.8 that these proteins have high quality alignments with template proteins. The results show that on this dataset, the performance of all the methods that utilize HTS scores improves. These improvements are quite large, ranging from 0.11 to 0.14 in terms of ROC, and from 0.14 to 0.25 in terms of PR. The best performance is achieved by the NN-HTS+PPS for ROC, and NN-HTS for PR scores. In contrast, the performance gains achieved by ISIS and NN-PSSM+SSE on DS2 are quite small, ranging from 0.025 to 0.030. Overall, these results indicate that the rather simple method developed in Section 5.2.7 can successfully identify the proteins for which the methods developed here make reliable predictions.

5.3.1 Binning Based on Length

In order to better understand the performance differences between the various schemes, we divide our results based on the length of the target protein. Figure 5.2 shows the average per-protein ROC scores for ten different subsets of DS1. These subsets are obtained by sorting the proteins in increasing order according to their length and then dividing them into ten equal size groups. Moving from left to right in this figure, each successive group contains proteins that are longer than the previous groups. For clarity, Figure 5.2 only shows the results obtained for ISIS, HTS, and SIRP.

These results show that ISIS performs better than both HTS and SIRP for the proteins in the first two groups–the shorter proteins–but performs worse for all but the last group, and then only when compared with HTS. As such, HTS-based methods lead to relatively lower quality predictions at the extreme ends of the protein length distribution. This is probably due to their reliance on high quality alignments to form predictions. Extreme target protein lengths decrease the chance that good template alignments will be found, as there are fewer templates of the proper size available.

5.4 Discussion

We have shown how combining HTS scores with profiles, secondary structure predictions, and per-residue alignment scores can be used to train a neural network to predict interacting residues. The novelty of this approach comes from the implicit incorporation of global sequence information by way of sequence alignments, as previous approaches focus on local features from sequence windows. In addition, we present a technique for measuring the accuracy of predictions made by our method, and leveraging this technique dramatically improves performance.

Chapter 6

Conclusion

In this dissertation we presented a number of algorithms that have advanced the state of the art in several areas of bioinformatics, though the implications extend beyond biological data. We focused on sequence alignments in Chapter 2, which are the foundation for countless bioinformatic applications. One of these applications is the identification of residues involved in protein-protein interaction, for which we developed an improved technique in Chapter 5. Sequence alignments also serve as the basis for various structure prediction techniques, and in Chapter 3 we have described an algorithm for optimizing the assembly of fragments into complete protein structures. Finally, when different structure prediction techniques provide different structures, our algorithm from Chapter 4 can determine their quality.

Even for reasonably short sequences, there are an astronomical number of possible alignments between two sequences under a single objective. With multiple objectives, the problem grows exponentially. However, using multiple objectives can produce better alignments. We proved that the problem of creating Pareto optimal frontiers of sequence alignments has optimal substructure. Using this property, we designed efficient algorithms for generating Pareto optimal frontiers of protein sequence alignments. Combining multiple objective functions in a Pareto optimal framework led to better alignments than were possible using a linear combination of objectives, or single objectives in isolation. We described algorithms for dramatically reducing the required time for generating accurate approximations of a Pareto optimal frontier of alignments. The alignment algorithm and optimizations presented here are independent of the type of

sequence and objective functions used.

Using high quality alignments, we developed a technique for predicting, from sequence alone, which residues interact with other proteins. Our method (SIRP) constructed induced multiple sequence alignments from pairwise alignments, then examined the labels from these alignments to vote on the likelihood of a residue interacting. We developed a method for assessing the confidence of predictions made by SIRP. This confidence estimating technique could be applied to most any prediction method that relies on alignments. When restricting ourselves to test cases where we can make confident predictions, SIRP substantially outperformed an existing method.

Moving from sequences to structures, we developed efficient deterministic methods for optimizing fragment assembly in the context of predicting protein structures. The efficiency of these methods was partly due to the efficient checking of steric conflicts. These methods outperformed an existing stochastic method both in terms of speed and the quality of the resulting structure, with the gains in performance growing as the size of the problem increases. Our methods created better structures given more time to run, while the performance of stochastic methods saturated.

We developed several techniques for assessing the quality of a predicted structure. These methods produce better assessments of predicted structure quality than Pcons, the previous top performing technique. Our consensus-based approach used data from existing quality predictors, so the method was designed to be flexible and easily retrained to handle a change in the set of predictors. We also developed a method for appropriately handling missing prediction values.

6.1 Directions for Future Research

This dissertation provides the basis of several interesting directions for future research. Concerning Pareto optimal alignments, selecting the best alignment on a frontier remains challenging, and a good algorithm for this would be very useful. 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 indicates the strength of the evolutionary signals in the profiles. This could be used to produce more sensitive algorithms for functional site identification, for remote homology detection, or for building better profiles.

If structures for the template sequences exist, each of the alignments on a Pareto optimal frontier can be input to a threading algorithm such as Modeller [106] to produce protein structure predictions. Frontiers from multiple templates could be combined for a more reliable prediction. The structure scores produced by Modeller can be used to select the best prediction, or if a single template is used, our model quality assessment algorithm could be employed. To reduce the computational cost, a single template could be selected using single-objective alignments before generating the Pareto optimal frontier.

Using our interaction prediction algorithm, an algorithm could be designed to assess the quality of predicted structures. Interacting residues must be exposed on the protein surface, or they cannot contact other proteins. If a structure prediction contains buried residues which are predicted to be interacting, this indicates a low quality structure. The alignment quality assessment technique described in Chapter 5 could also be used to weed out low quality sequence alignments.

With respect to protein fragment assembly, the optimization procedure is only as good as the fragments themselves, so algorithms to select diverse and representative fragments would produce better structures. The assembly algorithm could also be used to generate decoy structures for evaluating model quality assessment techniques. An interesting biological insight could be gained from trying to build structures with lower free energy than the native structure. The structures could then be constructed by chemists for verification. If such structures exist, it would mean there are evolutionary pressures that took precedent over the stability of the structures.

References

[1] Hiroaki Fukunishi, Osamu Watanabe, and Shoji Takada. On the hamiltonian replica exchange method for efficient sampling of biomolecular systems: Application to protein structure prediction. *The Journal of Chemical Physics*, 116(20):9058–9067, 2002.

[2] N. Krasnogor, B.P. Blackburne, E.K. Burke, and J.D. Hirst. Multimeme algorithms for protein structure prediction. In Juan Julian Merelo Guervs, Panagiotis Adamidis, Hans-Georg Beyer, Hans-Paul Schwefel, and Jose Luis Fernandez-Villacas, editors, *Parallel Problem Solving from Nature*, volume 2439 of *Lecture Notes in Computer Science*, pages 769–778. Springer Berlin Heidelberg, 2002.

[3] Yang Zhang. Progress and challenges in protein structure prediction. *Current Opinion in Structural Biology*, 18(3):342 – 348, 2008.

[4] David T Jones. Progress in protein structure prediction. *Current Opinion in Structural Biology*, 7(3):377 – 387, 1997.

[5] Frank Eisenhaber, Bengt Persson, and Patrick Argos. Protein structure prediction: Recognition of primary, secondary, and tertiary structural features from amino acid sequence. *Critical Reviews in Biochemistry and Molecular Biology*, 30(1):1–94, 1995, http://informahealthcare.com/doi/pdf/10.3109/10409239509085139. PMID: 7587278.

[6] C. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181:223–230, 1973.

[7] SF Altschul, TL Madden, AA Schaffer, J Zhang, Z Zhang, W Miller, and DJ Lipman. Gapped blast and psi-blast: A new generation of protein database search programs. *Nucl. Acids Res.*, 25(17):3389–3402, 1997, http://nar.oxfordjournals.org/cgi/reprint/25/17/3389.pdf.

[8] Ruslan Sadreyev and Nick Grishin. Compass: a tool for comparison of multiple protein alignments with assessment of statistical significance. *Journal of Molecular Biology*, 326(1):317–336, 2003.

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

[10] J D Thompson, F Plewniak, and O Poch. Balibase: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15(1):87–88, 1999.

[11] Akito Taneda. Multi-objective pairwise rna sequence alignment. *Bioinformatics*, 26(19):2383–2390, Oct 2010.

[12] Akito Taneda. An efficient genetic algorithm for structural rna pairwise alignment and its application to non-coding rna discovery in yeast. *BMC Bioinformatics*, 9:521, 2008.

[13] Pasut Seeluangsawat and Prabhas Chongstitvatana. A multiple objective evolutionary algorithm for multiple sequence alignment. *Proceedings of the 2005 conference on Genetic and evolutionary computation GECCO 05*, 1:477, 2005.

[14] Fernando Jos Mateus da Silva, Juan Manuel Snchez Prez, Juan Antonio Gmez Pulido, and Miguel A Vega Rodrguez. Parallel niche pareto alineaga–an evolutionary multiobjective approach on multiple sequence alignment. *J Integr Bioinform*, 8(3):174, 2011.

[15] M A Roytberg, M N Semionenkov, and O Iu Tabolina. Pareto-optimal alignment of biological sequences. *Biofizika*, 44(4):581–594, 1999.

[16] D. Gusfield, K. Balasubramanian, and D. Naor. Parametric optimization of sequence alignment. In *Proceedings of the third annual ACM-SIAM symposium*

on Discrete algorithms, SODA '92, pages 432–439, Philadelphia, PA, USA, 1992. Society for Industrial and Applied Mathematics.

[17] Lus Paquete and Joo P. O. Almeida. Experiments with bicriteria sequence alignment. In Yong Shi, Shouyang Wang, Yi Peng, Jianping Li, and Yong Zeng, editors, *Cutting-Edge Research Topics on Multiple Criteria Decision Making*, volume 35 of *Communications in Computer and Information Science*, pages 45–51. Springer Berlin Heidelberg, 2009.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182 –197, April 2002.

[19] C.M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[20] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multiobjective optimization. In Ajith Abraham, Lakhmi Jain, Robert Goldberg, Lakhmi C. Jain, and Xindong Wu, editors, *Evolutionary Multiobjective Optimization*, Advanced Information and Knowledge Processing, pages 105–145. Springer Berlin Heidelberg, 2005.

[21] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb. A simulated annealing-based multiobjective optimization algorithm: Amosa. *Evolutionary Computation, IEEE Transactions on*, 12(3):269 –283, June 2008.

[22] Zixing Cai and Yong Wang. A multiobjective optimization-based evolutionary algorithm for constrained optimization. *Evolutionary Computation, IEEE Transactions on*, 10(6):658 –675, December 2006.

[23] Aboubekeur Hamdi-Cherif Mohamed Ben Othman and Gamil A. Azim. Genetic algorithms and scalar product forpairwise sequence alignment. *International Journal of Computers*, 2(2), 2008.

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

[25] Cdric Notredame, Emmet A. O'Brien, and Desmond G. Higgins. Raga: Rna sequence alignment by genetic algorithm. *Nucleic Acids Research*, 25(22):4570–4580, 1997, http://nar.oxfordjournals.org/content/25/22/4570.full.pdf+html.

[26] Ching Zhang and A.K.C. 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*, 27(6):918 –932, December 1997.

[27] Antonio López Jaimes and Carlos A. Coello Coello. An introduction to multi-objective evolutionary algorithms and some of their potential uses in biology. In Tomasz G. Smolinski, Mariofanna G. Milanova, and Aboul Ella Hassanien, editors, *Applications of Computational Intelligence in Biology*, volume 122 of *Studies in Computational Intelligence*, pages 79–102. Springer, 2008.

[28] Mikkel T. Jensen. Reducing the run-time complexity of multiobjective eas: The nsga-ii and other algorithms. *IEEE Trans. Evolutionary Computation*, 7(5):503–515, 2003.

[29] Hongbing Fang, Qian Wang, Yi-Cheng Tu, and Mark F. Horstemeyer. An efficient non-dominated sorting method for evolutionary algorithms. *Evolutionary Computation*, 16(3):355–384, 2008.

[30] 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*, GECCO '07, pages 789–796, New York, NY, USA, 2007. ACM.

[31] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708, 1982.

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

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

[34] Arun S Konagurthu, James C Whisstock, Peter J Stuckey, and Arthur M Lesk. Mustang: a multiple structural alignment algorithm. *Proteins*, 64(3):559–574, Aug 2006.

[35] Carsten Kemena and Cedric Notredame. Upcoming challenges for multiple sequence alignment methods in the high-throughput era. *Bioinformatics*, 25(19):2455–2465, 2009.

[36] Hsin-Nan Lin, Cdric Notredame, Jia-Ming Chang, Ting-Yi Sung, and Wen-Lian Hsu. Improving the alignment quality of consistency based aligners with an evaluation function using synonymous protein words. *PLoS ONE*, 6(12):e27872, 12 2011.

[37] A. Heger and L. Holm. Picasso: generating a covering set of protein family profiles. *Bioinformatics*, 17(3):272–279, 2001.

[38] R. Hughey M. Cline and K. Karplus. Predicting reliable regions in protein sequence alignments. *Bioinformatics*, 18:306–314, 2002.

[39] Yongchao Liu, Bertil Schmidt, and Douglas L. Maskell. Msaprobs: multiple sequence alignment based on pair hidden markov models and partition function posterior prob abilities. *Bioinformatics*, 26(16):1958–1964, 2010.

[40] K. T. Simons, C. Kooperberg, E. Huang, and D. Baker. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and bayesian scoring functions. *Journal of Molecular Biology*, 268:209–225, 1997.

[41] K. Karplus, R. Karchin, J. Draper, J. Casper, Y. Mandel-Gutfreund, M. Diekhans, and R. Hughey. Combining local-structure, fold-recognition, and new fold methods for protein structure prediction. *PROTEINS: Structure, Function and Genetics*, 53:491–496, 2003.

[42] J. Lee, S. Kim, K. Joo, I. Kim, and J. Lee. Prediction of protein tertiary structure using profesy, a novel method based on fragment assembly and conformational space annealing. *PROTEINS: Structure, function and bioinformatics*, 56:704–714, 2004.

[43] C.M. Fiduccia and R.M. Mattheyses. A linear-time heuristic for improving network partitions. *Proceedings of the 19th Design Automation Conference*, pages 175–181, 1982.

[44] 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. *Journal of Molecular Biology*, 247:536–540, 1995.

[45] K. Karplus, S. Katzman, G. Shackleford, M. Koeva, J. Draper, B. Barnes, Marcia Soriano, and R. Hughey. Sam-t04: what's new in protein-structure prediction for casp6. *PROTEINS: Structure, Function and Bioinformatics*, 2005.

[46] J. U. Bowie and D. Eisenberg. An evolutionary approach to folding small alpha-helical proteins that uses sequence information and an empirical guiding fitness function. *Proceedings of the National Academy of Sciences of the United States of America*, 91:4436–4440, 1994.

[47] C. A. Rohl, C. E. M. Strauss, K. M. S. Misura, and D. Baker. Protein structure prediction using rosetta. *Methods in Enzymology*, 383:66–93, 2004.

[48] B. H. Park and M. Levitt. The complexity and accuracy of discrete state models of protein structure. *Journal of Molecular Biology*, 249:493–507, 1995.

[49] R. Kolodny, P. Koehl, L. Guibas, and M. Levitt. Small libraries of protein fragments model native protein structures accurately. *Journal of Molecular Biology*, 323:297–307, 2002.

[50] P. Tuffery and P. Derreumaux. Dependency between consecutive local conformations helps assemble protein structures from secondary structures using go potential and greedy algorithm. *Protein Science*, 61:732–740, 2005.

[51] P. Tuffery, F. Guyon, and P. Derreumaux. Improved greedy algorithm for protein structure reconstruction. *Journal of Computational Chemistry*, 26:506–513, 2005.

[52] J. Chandonia, G. Hon, N. S. Walker, L. Lo Conte, P. Koehl, M. Levitt, and S. E. Brenner. The astral compendium in 2004. *Nucleic Acids Research*, 32:D189–D192, 2004.

[53] S. F. Altschul, L. T. Madden, A. A. Schffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–402, 1997.

[54] D. Mittelman, R. Sadreyev, and N. Grishin. Probabilistic scoring measures for profile-profile comparison yield more accurate short seed alignments. *Bioinformatics*, 19(12):1531–1539, 2003.

[55] S. Henikoff and J. G. Henikoff. Amino acid subsitution matrices from protein blocks. *PNAS*, 89:10915–10919, 1992.

[56] G. Wang and R. L. Dunbrack JR. Scoring profile-to-profile sequence alignments. *Protein Science*, 13:1612–1626, 2004.

[57] M. Marti-Renom, M. Madhusudhan, and A. Sali. Alignment of protein sequences by their profiles. *Protein Science*, 13:1071–1087, 2004.

[58] H. Rangwala and G. Karypis. Profile based direct kernels for remote homology detection and fold recognition. *Bioinformatics*, 21:4239–4247, 2005.

[59] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[60] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[61] D. Cozzetto, A. Kryshtafovych, M. Ceriani, and A. Tramontano. Assessment of predictions in the model quality assessment category. *Proteins: Structure, Function and Bioinformatics*, 69(Suppl 8):175–183, 2007.

[62] B. Wallner and A. Elofsson. Can correct protein models be identified?. *Protein Science*, 12:1073–1086, 2003.

[63] S.C. Tosatto. The victor/frst function for model quality estimation. *Journal of Computational Biology*, 12:1316–1327, 2005.

[64] J Shi. Modeling and quality assessment using harmony3 and quad. *CASP7 Proceedings*, 1:118, 2006.

[65] B. Wallner and A. Elofsson. Identification of correct regions in protein models using structural, alignment, and consensus information. *Protein Science*, 4:900–913, 2006.

[66] H Chen, Y F Yang, and D Kihara. Quality assessment using the diversity of suboptimal alignments. *CASP7 Proceedings*, 1:25, 2006.

[67] M et al Boniecki. Identification and refinement of potential errors in protein structures. *CASP7 Proceedings*, 1:50–51, 2006.

[68] B. Wallner and A. Elofsson. Prediction of global and local model quality in casp7 using pcons and proq. *Proteins: Structure, Function, and Bioinformatics*, 69(S8):184–193, 2007.

[69] S. Cristobal, A. Zemla, D. Fischer, L. Rychlewski, and A. Elofsson. A study of quality measured for protein threading models. *BMC Bioinformatics*, 2(5), 2001.

[70] M. Levitt and M. Gerstein. A unified statistical framework for sequence comparison and structure comparison. *Proceedings of National Academy of Science, USA*, 95:5913–5920, 1998.

[71] A. Zemla. Lga: a method for finding 3d similarities in protein structures. *Nucleic Acids Research*, 31(13):3370–3374, 2003.

[72] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. *Proceedings of ACM E-Commerce*, 1:158–167, 2000.

[73] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. *WWW10 Proceedings*, 1:285–295, 2001.

[74] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22(1):143–177, 2004.

[75] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

[76] T. Joachims. *Advances in Kernel Methods: Support Vector Learning*, chapter Making large-Scale SVM Learning Practical. MIT-Press, 1999.

[77] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[78] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proc Natl Acad Sci U S A*, 98(8):4569–4574, Apr 2001.

[79] Jun Chen, Jianhong Zhou, Weon Bae, Claire K Sanders, John P Nolan, and Hong Cai. A yegfp-based reporter system for high-throughput yeast two-hybrid assay by flow cytometry. *Cytometry A*, 73(4):312–320, Apr 2008.

[80] L. Giot, J. S. Bader, C. Brouwer, A. Chaudhuri, B. Kuang, Y. Li, Y. L. Hao, C. E. Ooi, B. Godwin, E. Vitols, G. Vijayadamodar, P. Pochart, H. Machineni, M. Welsh, Y. Kong, B. Zerhusen, R. Malcolm, Z. Varrone, A. Collis, M. Minto, S. Burgess, L. McDaniel, E. Stimpson, F. Spriggs, J. Williams, K. Neurath, N. Ioime, M. Agee, E. Voss, K. Furtak, R. Renzulli, N. Aanensen, S. Carrolla, E. Bickelhaupt, Y. Lazovatsky, A. DaSilva, J. Zhong, C. A. Stanyon, R. L. Finley, K. P. White, M. Braverman, T. Jarvie, S. Gold, M. Leach, J. Knight, R. A. Shimkets, M. P. McKenna, J. Chant, and J. M. Rothberg. A protein interaction map of drosophila melanogaster. *Science*, 302(5651):1727–1736, Dec 2003.

[81] P. Uetz, L. Giot, G. Cagney, T. A. Mansfield, R. S. Judson, J. R. Knight, D. Lockshon, V. Narayan, M. Srinivasan, P. Pochart, A. Qureshi-Emili, Y. Li, B. Godwin, D. Conover, T. Kalbfleisch, G. Vijayadamodar, M. Yang, M. Johnston, S. Fields, and J. M. Rothberg. A comprehensive analysis of protein-protein interactions in saccharomyces cerevisiae. *Nature*, 403(6770):623–627, Feb 2000.

[82] Siming Li, Christopher M Armstrong, Nicolas Bertin, Hui Ge, Stuart Milstein, Mike Boxem, Pierre-Olivier Vidalain, Jing-Dong J Han, Alban Chesneau, Tong Hao, Debra S Goldberg, Ning Li, Monica Martinez, Jean-Franois Rual, Philippe Lamesch, Lai Xu, Muneesh Tewari, Sharyl L Wong, Lan V Zhang, Gabriel F Berriz, Laurent Jacotot, Philippe Vaglio, Jrme Reboul, Tomoko

Hirozane-Kishikawa, Qianru Li, Harrison W Gabel, Ahmed Elewa, Bridget Baumgartner, Debra J Rose, Haiyuan Yu, Stephanie Bosak, Reynaldo Sequerra, Andrew Fraser, Susan E Mango, William M Saxton, Susan Strome, Sander Van Den Heuvel, Fabio Piano, Jean Vandenhaute, Claude Sardet, Mark Gerstein, Lynn Doucette-Stamm, Kristin C Gunsalus, J. Wade Harper, Michael E Cusick, Frederick P Roth, David E Hill, and Marc Vidal. A map of the interactome network of the metazoan c. elegans. *Science*, 303(5657):540–543, Jan 2004.

[83] Ulrich Stelzl, Uwe Worm, Maciej Lalowski, Christian Haenig, Felix H Brembeck, Heike Goehler, Martin Stroedicke, Martina Zenkner, Anke Schoenherr, Susanne Koeppen, Jan Timm, Sascha Mintzlaff, Claudia Abraham, Nicole Bock, Silvia Kietzmann, Astrid Goedde, Engin Toksz, Anja Droege, Sylvia Krobitsch, Bernhard Korn, Walter Birchmeier, Hans Lehrach, and Erich E Wanker. A human protein-protein interaction network: a resource for annotating the proteome. *Cell*, 122(6):957–968, Sep 2005.

[84] Sjoerd J de Vries and Alexandre M J J Bonvin. How proteins get in touch: interface prediction in the study of biomolecular complexes. *Curr Protein Pept Sci*, 9(4):394–406, Aug 2008.

[85] Iakes Ezkurdia, Lisa Bartoli, Piero Fariselli, Rita Casadio, Alfonso Valencia, and Michael L Tress. Progress and challenges in predicting protein-protein interaction sites. *Brief Bioinform*, 10(3):233–246, May 2009.

[86] Huan-Xiang Zhou and Sanbo Qin. Interaction-site prediction for protein complexes: a critical assessment. *Bioinformatics*, 23(17):2203–2209, Sep 2007.

[87] Aleksey Porollo and Jarosaw Meller. Prediction-based fingerprints of protein-protein interactions. *Proteins*, 66(3):630–645, Feb 2007.

[88] H. X. Zhou and Y. Shan. Prediction of protein interaction sites from sequence profile and residue neighbor list. *Proteins*, 44(3):336–343, Aug 2001.

[89] Piero Fariselli, Florencio Pazos, Alfonso Valencia, and Rita Casadio. Prediction of protein–protein interaction sites in heterocomplexes with neural networks. *Eur J Biochem*, 269(5):1356–1361, Mar 2002.

[90] James R Bradford and David R Westhead. Improved prediction of protein-protein binding sites using a support vector machines approach. *Bioinformatics*, 21(8):1487–1494, Apr 2005.

[91] Jo-Lan Chung, Wei Wang, and Philip E Bourne. Exploiting sequence and structure homologs to identify protein-protein binding sites. *Proteins*, 62(3):630–640, Mar 2006.

[92] Huiling Chen and Huan-Xiang Zhou. Prediction of interface residues in protein-protein complexes by a consensus neural network method: test against nmr data. *Proteins*, 61(1):21–35, Oct 2005.

[93] Yanay Ofran and Burkhard Rost. Isis: interaction sites identified from sequence. *Bioinformatics*, 23(2):e13–e16, Jan 2007.

[94] I. Res, I. Mihalek, and O. Lichtarge. An evolution based classifier for prediction of protein interfaces without using protein structures. *Bioinformatics*, 21(10):2496–2501, May 2005.

[95] Asako Koike and Toshihisa Takagi. Prediction of protein-protein interaction sites using support vector machines. *Protein Eng Des Sel*, 17(2):165–173, Feb 2004.

[96] Chris Kauffman and George Karypis. Librus: combined machine learning and homology information for sequence-based ligand-binding residue prediction. *Bioinformatics*, 25(23):3099–3107, Dec 2009.

[97] Miho Higurashi, Takashi Ishida, and Kengo Kinoshita. Pisite: a database of protein interaction sites using multiple binding states in the pdb. *Nucleic Acids Res*, 37(Database issue):D360–D364, Jan 2009.

[98] George Karypis. Yasspp: Better kernels and coding schemes lead to improvements in svm-based secondary structure prediction. *Proteins: Structure, Function and Bioinformatics*, 64(3):575–586, 2006.

[99] David Mittelman, Ruslan Sadreyev, and Nick Grishin. Probabilistic scoring measures for profile-profile comparison yield more accurate short seed alignments. *Bioinformatics*, 19(12):1531–1539, Aug 2003.

[100] A. Heger and L. Holm. Picasso: generating a covering set of protein family profiles. *Bioinformatics*, 17(3):272–279, Mar 2001.

[101] Huzefa Rangwala and George Karypis. frmsdpred: predicting local rmsd between structural fragments using sequence information. *Comput Syst Bioinformatics Conf*, 6:311–322, 2007.

[102] Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, Jul 2006.

[103] Burkhard Rost, Guy Yachdav, and Jinfeng Liu. The predictprotein server. *Nucleic Acids Res*, 32(Web Server issue):W321–W326, Jul 2004.

[104] T. Fawcett. Roc graphs: Notes and practical considerations for researchers, 2004.

[105] J. D. Fischer, C. E. Mayer, and J. Söding. Prediction of protein functional residues from sequence by probability density estimation. *Bioinformatics*, Jan 2008.

[106] A. Fiser, R. K. Do, and A.Sali. Modeling of loops in protein structures. *Protein Science*, 9:1753 – 1773, 2000.

Appendix A

Glossary And Acronyms

Table A.1: Various notation used in this dissertation

Symbol(s)	Meaning
α	Scaling factor for simulated annealing schedule
Å	Angstrom (unit of measure equal to 10^{-10} meters)
A, B	Protein sequences
$A[i:j]$	Subsequence of protein A starting at a_i and ending at a_j
a_i	The ith residue of protein A
A_n	The length of protein n
BPO	Best performing objective
CASP	Critical assessment of structure prediction
CC	Pearson correlation coefficient (ρ): $\rho_{X,Y} = \frac{\sum_{i=1} n(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1} n(X_i - \bar{X})^2}\sqrt{\sum_{i=1} n(Y_i - \bar{Y})^2}}$
Cf	The Correlf objective
Cp	The Correlp objective
EA	Evolutionary algorithm
$E_{i,j}$	Element at row i and column j of matrix E
Ff	The Fdotf objective

Continued on next page

Table A.1 – continued from previous page

Symbol(s)	Meaning
FIM	Fraction improved
Fp	The Fdotp objective
ge	Gap extension penalty
go	Gap open penalty
HC	The hill climbing fragment assembly algorithm
HC_c	Hill-climbing optimization with coarse-grained locking
HC_f	Hill-climbing optimization with fine-grained locking
HTS	Homology transfer score
k	Window size for k-mer, also an index on an objective
m	Two-dimensional amino acid scoring matrix
$\mathbf{m}$	Multi-dimensional amino acid scoring matrix
MS	Match score of an alignment
n	Number of fragments in neighbor lists
NCBI	National Center for Biotechnology Information
OPW	Objective performance weighted
Pc	The Picasso objective
$PF(\mathcal{V})$	Operator to create a Pareto optimal subset from $\mathcal{V}$
PSFM	Position specific frequency matrix
PSSM	Position specific scoring matrix
r	Number of complete simulated annealing runs
Rf	The Rankf objective
RMSD	Root mean squared deviation
RMSE	Root mean squared error
Rp	The Rankp objective
SA	The simulated annealing fragment assembly algorithm
SCOP	Structural Classification of Proteins (database)
SS	The SS_dotp objective
$\vec{s_1}$	A solution to a multi-objective optimization problem
$s_1 \succ s_2$	Solution s_1 dominates s_2

Continued on next page

Table A.1 – continued from previous page

Symbol(s)	Meaning
V, F, E, G	Matrices
$\mathcal{V}, \mathcal{F}, \mathcal{E}, \mathcal{G}$	Matrices of sets
$\mathcal{V}$	Operator to subtract a vector from each element in a set $\mathcal{V}$