1-1-2011

# A Graph-Based Algorithm to Determine Protein Structure from Cryo-EM Data

Stephen Schuh
*Washington University in St. Louis*

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science

Department of Computer Science and Engineering

Thesis Examination Committee:
Tao Ju, Chair
Kunal Agrawal
Robert Pless

# A GRAPH-BASED ALGORITHM TO DETERMINE PROTEIN STRUCTURE FROM CRYO-EM DATA

by

Stephen Schuh

A thesis presented to the School of Engineering
of Washington University in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

May 2011
Saint Louis, Missouri

ABSTRACT OF THE THESIS

A Graph-Based Algorithm to Determine Protein Structure from Cryo-EM Data

by

Stephen Schuh

Master of Science in Computer Science

Washington University in St. Louis, 2011

Research Advisor: Professor Tao Ju

Cryo-electron microscopy (cryo-EM) provides 3D density maps of proteins, but these maps do not have sufficiently high resolution to directly yield atomic-scale models. Previous work has shown that features known as secondary structures can be located in these density maps. A second source of information about proteins is sequence analysis, which predicts locations of secondary structures along the protein sequence but does not provide any information about the 3D shape of the protein. This thesis presents a graph-based algorithm to find the correspondence between the secondary structures in the density map and sequence. This provides an ordering of secondary structures in the 3D density map, which can be used in building an atomic-scale model of the protein.

# Acknowledgments

I thank Tao for his contagious enthusiasm about computer graphics and for his support, encouragement, and guidance in helping me to conceptualize and complete this work.

I thank Sasakthi for all the work this thesis builds on and for his patience and kindness in helping me through this project.

I thank all my neighbors in the Media and Machines Lab for cheerful company and engaging conversations.

And I thank Anne for all good things outside of computer science.

Stephen Schuh

*Washington University in Saint Louis*
*May 2011*

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Proteins are essential parts of all organisms and are the chief actors within the cell. The nature of the interactions between proteins is determined in part by their 3D structure. Protein structure determination is of great interest to biologists.

There are a variety of ways to determine the 3D structure of a protein. Many methods take as input a 3D density map of the protein and produce as output an atomic model of the protein backbone. The density map is a 3D array of measurements of density of the structure at each location in space.

The three most common methods of obtaining density maps are x-ray diffraction, nuclear magnetic resonance (NMR) spectroscopy, and cryo-electron microscopy (cryo-EM). Each method has advantages and disadvantages: x-ray diffraction and NMR tend to yield higher-resolution density maps, while cryo-EM allows observation of larger protein complexes and enables measurement of samples in their natural environments. A survey of the structures stored in the Protein Data Bank [14] as of the end of 2010 shows that x-ray diffraction has produced 87% of structures, NMR has produced 12%, and cryo-EM has produced fewer than 0.5% [16].

Given a density map, how can a 3D model of the protein backbone be created? If the input density map has sufficiently high resolution to discern the shape of the protein backbone, a relatively simple approach is to segment out the high-density backbone from the density map.

But what if the resolution of the density map is not high enough to directly resolve the backbone? This thesis presents a method of solving this problem by locating larger-scale features in the density map and determinining how best to connect those larger-scale features together to form an estimate of the protein backbone.

## 1.1 Overview

The state of the art in cryo-EM based single particle reconstruction [34] provides density volumes at resolutions from four to ten Angstroms, and thus cannot be directly used to determine the locations of amino acid residues.

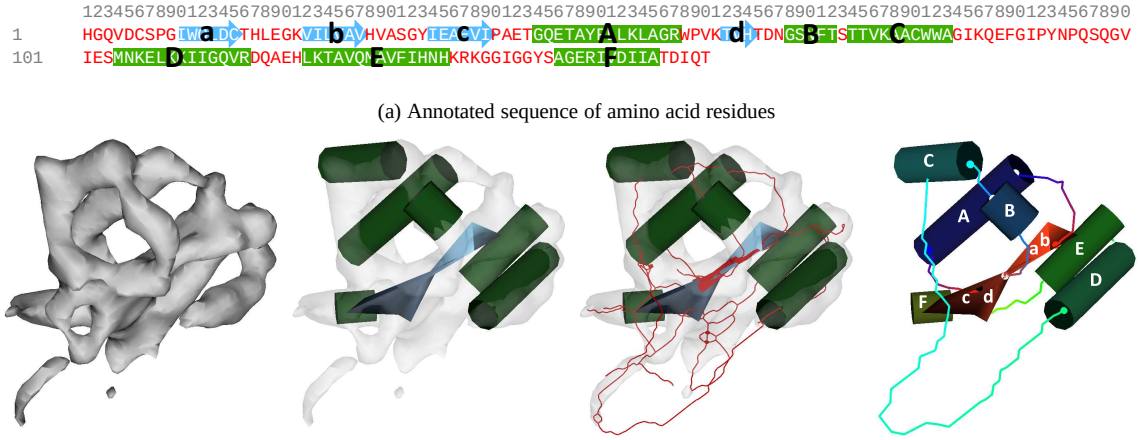
However, as seen in Figure 1.1b, secondary structure elements are easily observed at these resolutions due to their characteristic tubular and plate-like shapes. This has led to the development of many manual [35] and automatic techniques such as SSEHunter [4], HelixHunter [19], SheetMinter [22] and SheetTracer [23], which use geometric skeletons, template-based cross correlation and heuristics to locate the *observed* SSEs within the density volume. Figure 1.1c displays the results of one such method (SSEHunter). A survey of methods for detecting secondary structure elements (SSEs) in cryo-EM density volumes is provided by Chiu et al. [9].

With the advent of modern, large-scale DNA sequencing efforts such as the Human Genome Project [26], obtaining the sequence of amino acid residues of a protein has become a very accurate and efficient task. Subsequently, techniques such as PSIPred [20], JPred [11] (Figure 1.1a), Scratch [8] and many others have been developed to accurately and efficiently *predict* which amino acid residues in the sequence might form SSEs.

We present a method of bringing together knowledge about the *observed* SSEs in the 3D shape of a protein and the *predicted* SSEs in its amino acid sequence. We show how this enables the creation of an initial 3D shape of the protein backbone that can be refined by later steps in a model-building pipeline. We present here an extension of previous work from our group [1]; the main new contribution is the addition of $\beta$-sheets into the method. Additional contributions include implementation improvements and user interface changes.

## 1.2 Problem Statement

The computational problem that we address is the *correspondence* between the SSEs predicted from the sequence, and the ones observed in the density volume. As illustrated in Figure 1.1e, such a correspondence establishes a coarse 3D protein structure consisting of a chain of helices and sheets. It is important to note that this correspondence may not be a




(a) Annotated sequence of amino acid residues

(b) Density volume (c) Secondary structure elements (d) Geometric skeleton (e) SSE Correspondence

**Figure 1.1:** The inputs to our method are (a) the protein sequence with locations of $\alpha$-helices (green) and $\beta$-strands (blue) predicted using JPred [11]; (b) the 3D volume obtained by cryo-EM; (c) possible locations of SSEs in the 3D volume detected using SSEHunter [4]; and (d) the geometric skeleton computed from the density volume. (e) The correspondence between the SSEs in the sequence and the 3D volume, computed by our method.

bijection. Due to noise in a typical density volume, an SSE detection algorithm may fail to find the locations of all the SSEs within that volume and may also identify false SSEs.

The SSE correspondence problem has previously been studied in the work of Wu et al. [33] and in our earlier work [1]. Wu employed an exhaustive combinatorial search to find, amongst all permutations of SSEs in the density volume, an ordering that best matches the protein sequence. This brute-force algorithm has a factorial time complexity. According to their experiments, this method is only practical for very small inputs, taking 1.5 hours and 16 hours to find the correspondence of a 3-helix and 8-helix protein respectively. In the first version of our work [1] we achieved much better performance (i.e. 5 seconds for a 20-helix protein) by formulating the correspondence problem as a subgraph isomorphism. Because our previous method found correspondences for $\alpha$-helices only, it could not be used to generate accurate pseudo-backbones for proteins containing $\beta$-sheets.

## 1.3 Method

The key idea behind our method is to represent the density map and the sequence in a common way, and then step by step build up a correspondence between these two representations.

Our common representation is a graph, with nodes representing secondary structures and edges representing connectivity between secondary structures. We build the *sequence graph* by analyzing the sequence of amino acid residues, predicting positions of SSEs in the sequence, and connecting them together to form a sparse and linear graph. The *density map graph* is constructed by analyzing the observed SSEs in the density volume, and by using the geometric skeleton to identify their possible connectivity. Due to noise and the lack of high resolution in cryo-EM densities, the geometric skeleton may contain many alternate paths; therefore, this graph is often densely connected. Section 2.2 describes in detail how each graph is constructed.

After constructing the two graphs, the next task is to find the best correspondence between them. In other words, we seek the best mapping of the protein sequence graph onto the density map graph. The mapping must be robust to errors in the graphs such as missing SSEs or missing or extra connectivity between SSEs. With our graph formulation, this can be recast as the constrained, error-correcting graph-matching problem which seeks the best-matching simple paths along the two graphs.

To this problem we apply the best-first search algorithm, a popular method in graph matching problems. As required by the best-first algorithm, we design an SSE attribute-based cost function that assigns lower cost to more likely correspondences. This means that the first results returned by the best-first search have the globally minimal costs [25]. Section 2.4 describes this search and associated cost functions in detail.

We apply our method to a collection of authentic and simulated cryo-EM test data and show that it identifies the correct SSE correspondence with little or no user intervention for small and medium size proteins. For example, Figure 1.1e shows the correspondence computed by our method for the 2ITG protein of the Human Immunodeficiency Virus (HIV). Our approach improves the efficiency of an otherwise exhaustive search [33] by several orders of magnitude, obtaining the correspondence of proteins with more than 25 SSEs in under 40 seconds. In addition, the availability of the skeleton allows us to plot a path on the skeleton that connects successive SSEs, suggesting a possible pseudo-backbone of amino acid residues.

## 1.4 Contributions

In summary, we present the following contributions:

- We introduce a common representation of protein sequences and density volumes as attributed relational graphs, which is suitable for structural matching.
- We formulate a constrained error-correcting matching problem between attributed graphs, which differs from previously known exact and inexact matching problems. In addition we develop an optimal solution based on a best-first search.
- We present a novel and efficient computational approach for solving an open problem in structural biology, achieving orders of magnitude speedup over the best available method and making model building from cryo-EM volumes much easier for medium-size proteins.

## 1.5 Previous Work

**Graph matching** In pattern recognition and machine vision, graphs have long been used to represent object models such that object recognition reduces to graph matching. Here we briefly review graph matching problems and methodologies; more information about the wide variety of matching techniques is provided by the surveys of Bunke and Messmer [7] and Conte et al. [12].

In general, graph matching problems can be divided into exact matching and inexact matching. Exact matching aims at identifying a correspondence between a model graph and (a part of) an input graph, which can be solved using sub-graph isomorphism [30, 13] or graph monomorphism [32]. Because real-world data is seldom perfect and noise-free, inexact or error-correcting matching is desired in a large number of applications. As in the work of Bunke [5], error-correcting matching can be formulated as finding the bijection between two subgraphs from the model and input graph that minimizes some error function. This error typically consists of the cost of deforming the original graphs to their subgraphs and the error of matching the attributes of corresponding elements in the two subgraphs. Note that in most applications, the topology of the optimally matching subgraphs (e.g., whether it is connected, a tree, a path, etc.) is generally unknown. Such matching is said to be *un-constrained* since the minimization of the error function is the only goal.

The most popular algorithms for error-correcting graph matching are based on best-first and A* searches [25]. These algorithms are optimal in the sense that they are guaranteed to find the global optimal match. However, since the graph matching problem is NP-complete, the computational cost can be prohibitive for large graphs. To this end, various types of heuristic functions have been developed to prune the search space [29, 28, 6, 27, 32]. Other methods such as simulated annealing [17], neural networks [15], probabilistic relaxation [10], genetic algorithms [31], and graph decomposition [24] can also be used to reduce the computational cost. All these optimization methods are developed for un-constrained matching where the matched subgraphs can assume any topology.

For our problem, we know that the sequence is always a linear chain of connected secondary structure elements. We can use this observation to develop a specialized form of subgraph isomorphism that benefits from this reduced search space.

# Chapter 2

# Methods

## 2.1 Inputs: Density Map and Sequence

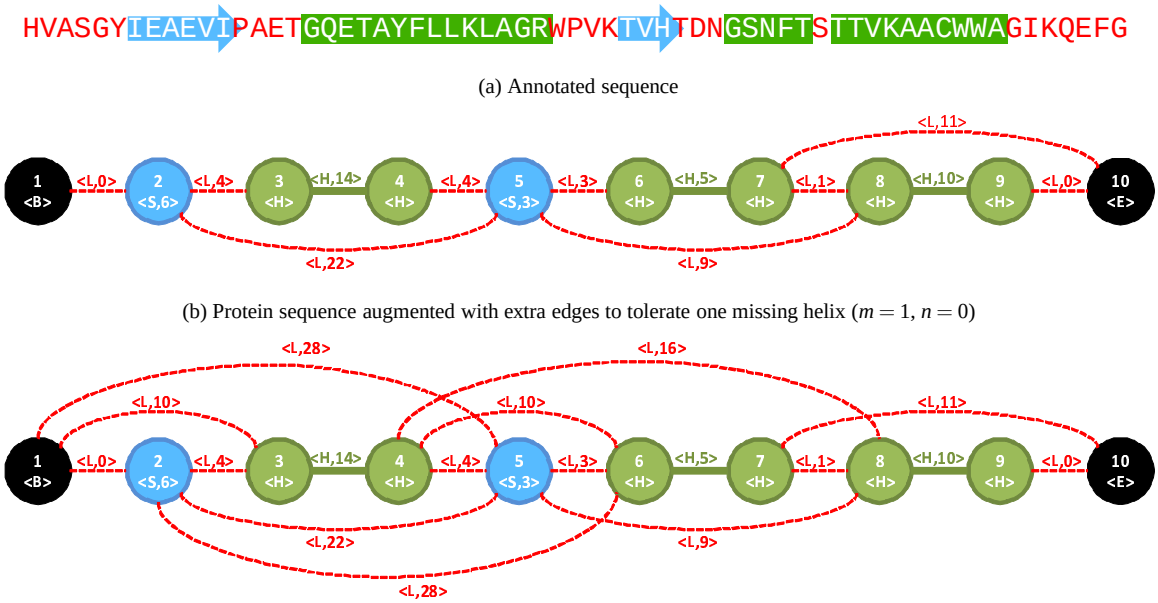
Our method takes inputs from two sources. The first is a cryo-EM density map with predicted $\alpha$-helix and $\beta$-sheet positions provided by SSEHunter and SSEBuilder. The second is a predicted amino acid sequence with predicted $\alpha$-helix and $\beta$-strand locations.

## 2.2 Graph Representation of Density Map and Sequence

We begin by representing the density map and the sequence as two graphs. In general, nodes in these graphs represent secondary structures and edges demonstrate possible connectivity between secondary structures. Figures 2.1 and 2.2 show examples of these graphs for the 2ITG protein of the HIV virus. The sections that follow describe our method of constructing these graphs.

### 2.2.1 Protein Sequence Graph

We represent the $\alpha$-helices and $\beta$-strands in the primary sequence using a collection of vertices and edges in the protein sequence graph. We represent each helix by two vertices connected by an edge, and we represent each strand as a single vertex. (This choice of two vertices per helix and one per strand is motivated by the density map graph, described in Section 2.2.2 below.) Each vertex has two associated parameters: $\alpha_{S1}$ represents the vertex type and is equal to $H$ or $S$ for helix or strand, respecitvely; $\alpha_{S2}$ represents the weight of




(a) Annotated sequence

(b) Protein sequence augmented with extra edges to tolerate one missing helix ($m = 1$, $n = 0$)

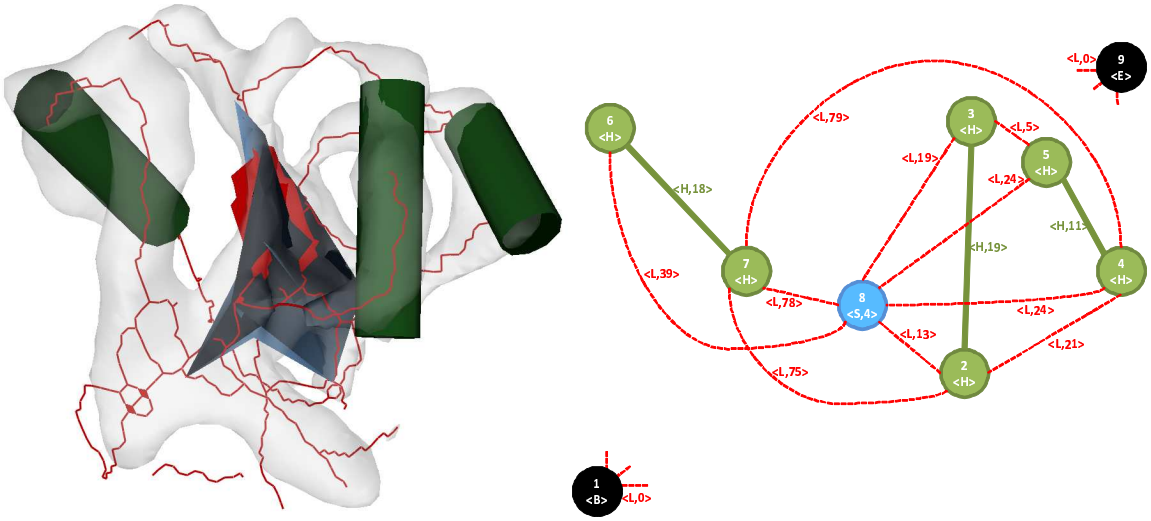
(c) Protein sequence augmented with extra edges to tolerate one missing helix and one missing sheet ($m = 1$, $n = 1$)

**Figure 2.1:** (a) The sequence of amino acid residues making up the 2ITG protein of the HIV virus and (b) the corresponding graphs designed to tolerate up to 1 missing sheet or (c) one missing helix and one missing sheet, where the vertices and edges have been colored by their attributes. Portions of the sequence have been omitted for simplicity; the full sequence is shown in Figure 1.1a.

the vertex. Strand vertices have weight equal to the number of amino acids in the strand; helix vertices have no weight.

We encode the lengths and connectivity of helices and strands using graph edges. We add edges to connect successive vertices in a linear fashion, forming a linear chain to represent the sequence. If an edge connects the two nodes representing a helix, that edge represents the helix body. All other edges represent the connections between neighboring helices and strands along the sequence. Each edge has two associated parameters: $\beta_{S1}$ is the edge type and is equal to $H$ for helix edges and $L$ otherwise; $\beta_{S2}$ is the edge weight, equal to the number of amino acids in the portion of the sequence represented by that edge.

This sequence graph will later be compared to a second graph representing the density map. We next modify the *sequence* graph to accommodate possible deficiencies in the *density map*, as follows. To allow for a missing $\alpha$-helix in the density map, we add extra edges bypassing each $\alpha$-helix in the sequence. Each of these edges has type $L$ and has a weight equal to the number of amino acids it bypasses. These edges create a path from beginning to end of the sequence bypassing one helix. Likewise, to allow for a missing




(a) Volume, skeleton and SSEs

(b) Density volume graph

**Figure 2.2:** (a) The density volume, skeleton, and detected SSEs, and (b) the corresponding graph, where the two terminal vertices 1 and 9 are connected to every other vertex via loop edges. Three helices have been omitted for simplicity; the full graph is shown in Figure 1.1d.

$\beta$-sheet in the density map, we add an extra edge bypassing each $\beta$-strand. Each of these bypass edges has type $L$ and weight equal to the number of amino acids it bypasses. After these additional edges are added, there is no longer a single path from the beginning to the end of the sequence graph; each possible path represents the complete sequence with zero or more missing helices or strands. In our implementation, the user specifies how many helices and sheets may be missing in the density map, and based on this input, an appropriate number of extra loops are added to the sequence graph.

Figure 2.1b shows the sequence graph including extra loops to bypass one helix, and Figure 2.1c shows a graph with extra loops to bypass one helix or one strand.

### 2.2.2 Density Volume Graph

As in the sequence graph, the volume graph $C$ consists of vertices and edges representing the secondary structures and the connections between them. Each $\alpha$-helix is represented by two vertices connected by an edge, and each $\beta$-sheet is represented by a single vertex. Each helix is represented by two vertices in order to encode the entry and exit points of the protein sequence as it passes through the helix. By contrast, each sheet is represented by a single vertex because one sheet corresponds to zero or more strands in the sequence,

and the number of strands per sheet cannot be known when the density volume graph is constructed. Each vertex has two parameters. $\alpha_{C1}$ represents the vertex type, which is $H$ for a helix vertex and $S$ for a sheet vertex. $\alpha_{C2}$ represents the vertex weight. Sheet nodes have weight equal to the expected strand length of that sheet, which is estimated based on the size of the sheet. Helix nodes have no weight.

We encode the connectivity between helices and sheets with graph edges. As in the sequence graph, each edge has a type $\beta_{C1}$, with $H$ for helix edges and $L$ for loop edges; a weight $\beta_{C2}$ representing the number of amino acids represented by that edge.

Like the sequence graph, the density map graph has one edge connecting the two vertices that represent each helix. This edge has type $H$ and weight equal to the estimated number of amino acids in that helix, computed based on the Euclidean distance between helix endpoints.

Unlike the sequence, the density map does not explicitly provide the needed connectivity between helices and sheets. To estimate the connectivity, we observe that secondary structures in the density map are likely to be connected in 3D through regions of high density in the map. We then seek a representation that depicts the topology of such high-density regions. To this end, we extract a morphological *skeleton* of the density using a combination of erosion-based binary [21] and grayscale [2] skeletonization techniques. Such skeletons can be robustly generated even from noisy surfaces while preserving the solid topology; an example is shown in Figure 2.2a.

Given the skeleton, we add an edge between every two vertices that are connected by a path along the skeleton, as long as that path does not pass through a helix. These edges have type $L$ and weight equal to the estimated number of amino acids along that path, computed based on the shortest-path distance along the skeleton.

We observe that due to noise in the input density map, the skeleton sometimes does not capture all the necessary connectivity among structures. For this reason we additionally add edges between any nodes that are within some user-specified distance of each other; if the skeleton captures the true connectivity this distance may be small; if the skeleton is sparse this threshold must be large. Edges added in this way have type $L$ and weight equal to the straight-line distance between nodes, expressed in units of number of amino acids.

We now highlight a key difference between our sequence graph and our density map, the representation of $\beta$-sheets. In the density graph one node represents an entire $\beta$-sheet,

whereas the sequence graph contains one node per $\beta$-strand. Since one sheet contains many strands, there will necessarily be a one-to-many correspondence between sheet nodes in the density map graph and strand nodes in the sequence graph. To accommodate this we add self-loops to the density map graph at every sheet node. These edges have type $L$ and weight equal to a user-specified number of amino acids, typically set to 5.

Finally we augment the density map by adding two terminal vertices of types $B$ and $E$. These vertices are virtual since we cannot predict the physical locations of the sequence end points based on the input density. We complete the graph by adding a loop edge from each terminal vertex to every other vertex, with type $L$ and no weight.

Figure 2.1b shows a density map graph computed by our method.

## 2.3 Graph Summary

In summary, we build two graphs, one representing the sequence and the other representing the density map. The graphs consist of vertices and edges, each having types and weights:

- Vertices:
  - Helix terminus: A helix terminus vertex has type $H$ and no weight.
  - Strand: A strand node has type $S$ and weight equal to the number of amino acids in the strand.
  - Sheet: A sheet node has type $S$ and weight equal to the expected strand length in the sheet, estimated based on the sheet size.
  - Terminal: The nodes representing the beginning and end of the sequence have types $B$ and $E$, respectively, and no weight.
- Edges:
  - Helix edge: A helix edge has type $H$ and weight equal to the number of amino acids in the helix.
  - Loop edge: A loop edge has type $L$ and weight equal to the number of amino acids in the loop, computed by traversing the sequence or estimated based on the shortest distance between nodes along the skeleton.

## 2.4 Constrained Graph Matching

After building the graphs described in the previous sections, the problem of finding a good correspondence between a density map and a sequence can be recast as the problem of finding a good correspondences between the two graphs. We next define the graph matching problem, describe what we mean by a good correspondence, and present our method of finding such a correspondence.

### 2.4.1 Graph Matching

Given two graphs representing the secondary structure elements (helices and strands/sheets) in the sequence and in the volume, we show that finding the correspondence between the two sets of structures reduces to a constrained graph matching problem. We begin by defining a chain:

A *chain* of a graph $G$ is a sequence of nodes $\{v_1,\ldots,v_n\} \subseteq V_G$ that form a path in $G$. A chain is *ordered* if $v_1 = 1, v_n = |V_G|$, and $v_i < v_{i+1}$ for all $i \in [1,n-1]$. A chain is *simple* if $v_i \neq v_j$ for all $i,j \in [1,n-1]$.

For example, an ordered chain in the sequence graph consists of a sequence of nodes and edges depicting a linked sequence of helices and strands. A correspondence between structures in the sequence and the density map is therefore a bijection between an ordered, simple chain in the sequence graph and a chain in the density map graph. Note that the definition of a chain allows a *partial* correspondence between a subset of the structures in the sequence and the volume. The correspondence problem can be defined generally for any pair of attributed relational graphs:

**Correspondence Problem** Let $S,C$ be two ARGs. The correspondence problem is to find an ordered, simple chain $\{p_1,\ldots,p_n\} \subseteq V_S$ and chain $\{q_1,\ldots,q_n\} \subseteq V_C$ that minimize the matching cost:

$$\sum_{i=1}^{n} c_v(p_i,q_i) + \sum_{i=1}^{n-1} c_e(p_i,p_{i+1},q_i,q_{i+1}) \tag{2.1}$$

where $c_v, c_e$ are any given functions evaluating the cost of matching node $p_i$ with $q_i$ or edge $\{p_i,p_{i+1}\}$ with $\{q_i,q_{i+1}\}$.

Compared to graph matching problems such as exact graph (or subgraph) isomorphisms, inexact graph matching, and maximum common subgraph problems [18], the correspondence problem described here is unique in that it seeks best-matching subgraphs from two graphs that have a particular shape. Given such constraints, previous graph matching algorithms that are guided only by error-minimization can not be directly applied.

### 2.4.2 Cost Functions

We next explain our choice for the two cost functions $c_v, c_e$ in Equation 2.1 when matching the sequence graph and the volume graph. Note that the algorithm we present in the next section works for any non-negative cost function.

The two cost functions measure the similarity of the attributes associated with a pair of vertices or a pair of edges. The vertex cost function has two purposes: it ensures that two matched vertices are of the same type, and for a strand-sheet vertex pair, it computes the difference between the length of the strand and the expected strand length for that sheet. The vertex cost function is defined as:

$$c_v(x,y) = \begin{cases} |\alpha_{S_2} - \alpha_{C_2}|, & \text{if } \alpha_{S_1}(x) = \alpha_{C_1}(y) = \text{`S'} \\ 0, & \text{if } \alpha_{S_1}(x) = \alpha_{C_1}(y) \neq \text{`S'} \\ \infty, & \text{otherwise} \end{cases} \tag{2.2}$$

The edge cost function enforces type matching and computes the length difference between two helix edges or two loop edges, and is defined as:

$$c_e(x,y,u,v) = \begin{cases} |\beta_{S_2}(x,y) - \beta_{C_2}(u,v)|, & \text{if } \beta_{S_1}(x,y) = \beta_{C_1}(u,v), \\ & \text{and } y = x+1. \\ |\beta_{S_2}(x,y) - \beta_{C_2}(u,v)| + \gamma_S(x,y), & \text{if } \beta_{S_1}(x,y) = \beta_{C_1}(u,v), \\ & \text{and } y > x+1. \\ \infty, & \text{otherwise.} \end{cases} \tag{2.3}$$

Here, the $\gamma_S$ term penalizes missing helices and sheets in the volume graph and is set to be a weighted sum of the length of helices and strands bypassed by a link edge. For a link

edge in the protein sequence connecting nodes $x$ and $y$, we compute the penalty as:

$$\gamma_S(x,y) = \omega_h \sum_{\substack{x<i<y-1, \text{ and} \\ \beta_{S_1}(i,i+1) = \text{'H'}}} \beta_{S_2}(i,i+1) \quad + \quad \omega_s \sum_{\substack{x<i<y, \text{ and} \\ \alpha_{S_1}(i) = \text{'S'}}} \alpha_{S_2}(i) \tag{2.4}$$

where $\omega_h$ and $\omega_s$ are user-specified weights that adjust the influence of missing helices and missing strands in this penalty term.

### 2.4.3 An Optimal Best-First Search Algorithm

In this section, we present a best-first search algorithm for solving the correspondence problem defined above. Our method extends the tree-search method commonly applied to unconstrained error-correcting graph matching problems, and is guaranteed to find the optimal match.

To find a match between two graphs, a tree-search algorithm starts out from an initial, incomplete match and incrementally builds more complete matches. To find matching chains in graphs $S, C$, we first consider a partial match as a sequence of node-pairs

$$M_k = \{\{p_1, q_1\}, \ldots, \{p_k, q_k\}\}$$

where $\{p_1, \ldots, p_k\}$ and $\{q_1, \ldots, q_k\}$ are the initial portion of some ordered, simple chain in $S$ and some chain in $C$. Based on the definition of chains and our matching goal of minimizing cost functions, elements of $M_k$ must satisfy the following requirements:

- **Vertex requirement**: For all $i \in [1,k]$:

  $$p_1 = 1, \quad p_i \in V_S, \quad q_i \in V_C, \quad \text{and} \quad c_v(p_i, q_i) \neq \infty,$$

  and for all $j \in [1,k], \quad i \neq j, \quad \alpha_{C_1}(j) \neq$ 'S':

  $$q_i \neq q_j.$$

  In other words, the only vertices that may repeat in $M_k$ are sheet vertices in the volume graph, and vertices in each pair must be of the same type.

- **Edge requirement**: For all $i \in [1, k-1]$:

  $$p_i < p_{i+1}, \quad \{p_i, p_{i+1}\} \in E_S, \quad \{q_i, q_{i+1}\} \in E_C, \quad \text{and} \quad c_e(p_i, p_{i+1}, q_i, q_{i+1}) \neq \infty.$$

  In other words, $\{p_1, \ldots, p_k\}$ must form an ordered chain, and the two edges connecting the two nodes in neighboring pairs in $M_k$ must be of a same type.

Starting with an empty match $M_0 = \emptyset$, the search algorithm incrementally builds longer matching chains. Specifically, we define an *expansion* of a partial match $M_k$ as a new partial match $M_{k+1} = M_k \cup \{\{p_{k+1}, q_{k+1}\}\}$ such that the added nodes $p_{k+1}, q_{k+1}$ satisfy the node requirement and the added edges $\{p_k, p_{k+1}\}, \{q_k, q_{k+1}\}$ (for $k > 0$) satisfy the edge requirement. Note that usually a $M_k$ can be expanded into multiple $M_{k+1}$. A match $M_k$ is *complete* (i.e., no more expansion can be done) if $p_k = |V_S|$.

Observe that the search procedure essentially builds a tree structure with $M_0$ at the root of the tree, expanded partial matches $M_k$ at the $k$th level of the tree, and complete matches at the tree leaves. Our goal is to find the complete match that minimizes the matching error defined by Equation 2.1.

### 2.4.4 Best-First Search

To find the optimal match without performing an exhaustive tree search, we adopt the best-first search algorithm, which prioritizes the expansion of incomplete matches using the cost function. The best-first search algorithm works by maintaining all un-expanded partial matches in a priority queue and only expanding the partial match with the best (smallest) cost function value. Because the lowest-cost node is always expanded at every step, the first complete match is guaranteed to have the lowest cost of all possible matches. In our implementation we continue expanding in the best-first sense after the first complete match is found. The next complete match is guaranteed to have the second-lowest cost, and so on.

# Chapter 3

# User Interface

The algorithm desribed in Chapter 2 has been implemented in C++ and a user interface has been implemented using Python and Qt. This is included in the Gorgon project [3]. The following sections highlight features of the user interface by walking through the steps required to run the algorithm on the 1IRK protein.

The correspondence search is launched from Gorgon by selecting "Find SSE Correspondences..." from the Secondary Structure Element section of the Actions menu. This launches a dock item containing the user interface for the algorithm. The UI contains five tabs that correspond to the steps of using the algorithm. The leftmost tab contains prompts for the input files needed by the algorithm. A screenshot of the data sources tab is provided in Figure 3.1.

## 3.1 Loading Files

To specify input files for the algorithm, the user clicks on the appropriate button in the data sources tab, shown in Figure 3.1. The four input files required by the algorithm are described in Table 3.1. The file input tab also allows the user to specify files via a Settings file, which is described in detail in Section 3.6.

As the user specifies the filenames for the input data, the skeleton, $\alpha$-helices, and $\beta$-sheets are rendered in the main Gorgon pane. The skeleton is colored red by default, helices are gray, and sheets are green.

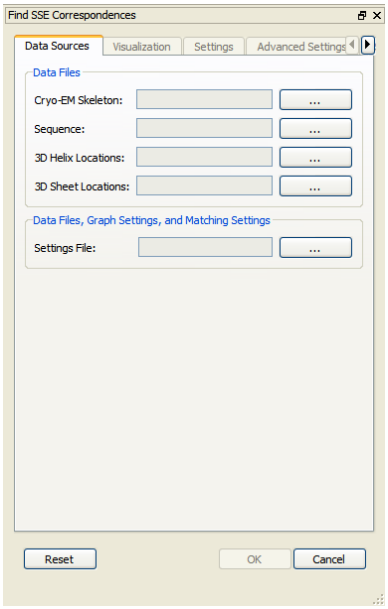



**Figure 3.1:** User interface for specifying input files

| Input type | Allowed file formats | Description |
|---|---|---|
| Cryo-EM Skeleton | Volume (.off or .mrc) or Mesh (.atom) | A skeleton derived from a CryoEM density map, consisting of 3D curves representing loops and $\alpha$-helices and 2D surfaces representing $\beta$-sheets. A skeleton would typically have been created earlier in the model-building process as part of the SSE Hunter algorithm. |
| Sequence | Sequence with SSE predictions (.seq) or full atomic model (.pdb) | A list of the amino acids of the structure with predicted locations of $\alpha$-helices and $\beta$-sheets |
| 3D Helix Locations | VRML file (.vrml, .wrl) or SSE Hunter output (.sse) | Locations and sizes of $\alpha$-helices in 3D density map, typically provided by the SSE Hunter algorithm |
| 3D Sheet Locations | VRML file (.vrml, .wrl) or SSE Hunter output (.sse) | Collections of triangles representing $\beta$-sheets in 3D density map, typically provided by the SSE Hunter algorithm |

**Table 3.1:** Input files required for the correspondence search algorithm

## 3.2 Visualizing Input Data and Graphs

After all the input files have been specified, the algorithm creates graphs representing the density map and the sequence and the Visualization tab is selected. Figure 3.2 shows the rendered skeleton, $\alpha$-helices, and $\beta$-sheets along with the Visualization controls.

The Visualization pane allows the user to hide or show the skeleton, helices, and sheets. The available options are described in Table 3.2.

At this point it is helpful to check that the $\beta$-sheets from the SSEHunter algorithm were correctly mapped onto the skeleton. This can be done by enabling “Show Skeleton Sheets”

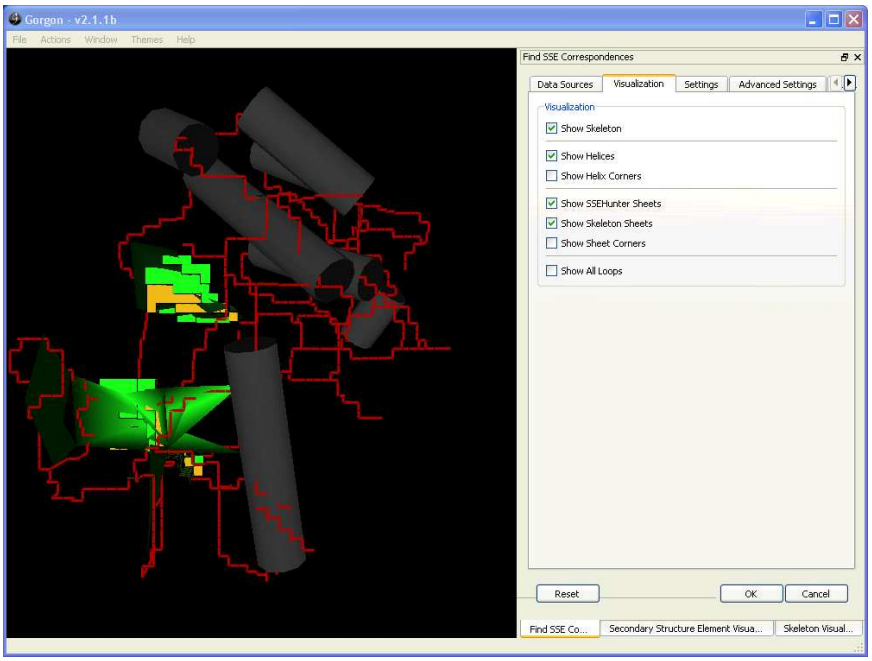



**Figure 3.2:** User interface for visualization options

| Visualization Option | Description |
|---|---|
| Show Skeleton | Show or hide the skeleton. |
| Show Helices | Show or hide the $\alpha$-helices. |
| Show SSEHunter Sheets | Show or hide the $\beta$-sheets provided as input to the algorithm. |
| Show Skeleton Sheets | Show or hide the $\beta$-sheets used by the matching algorithm. These are surfaces on the skeleton that are within a user-specified distance of an input $\beta$-sheet. |
| Show Sheet Corners | Show or hide the boundaries where $\beta$-sheets meet the skeleton curves. |
| Show All Loops | Show or hide the edges in the graph connecting one secondary structure to another. |

**Table 3.2:** Visualization options for the correspondence search

and visually comparing the yellow sheets on the skeleton to the green sheets from SSE-Hunter. If the skeleton sheets are much smaller or larger than the SSEHunter sheets, the user may decide to generate a new skeleton with sheets of the appropriate size.

It is also useful to examine the connectivity of the density map graph. The loops in the graph can be visualized by enabling “Show All Loops” and disabling “Show Skeleton”. If there are very few loops (in other words, if there are few edges between graph nodes), the true connectivity of the structure may not be represented by the graph. To correct this, the user can create a new skeleton with a lower density threshold, refine the skeleton using the grayscale skeletonization method, or add Euclidean edges (see Section 3.5). If there are very many loops (in other words, if the graph is fully connected) the correspondence

algorithm may run out of memory searching for the best correspondence. The number of loops can be reduced by creating a new skeleton with a higher density threshold.

The next step after visualizing the graphs is to specify the parameters for the algorithm and run the correspondence search.

## 3.3 Computing Correspondences and Viewing Results

The correspondence search relies on several parameters that determine graph weights and cost funtion properties. The correspondence algorithm can be run with default values for all these parameters by clicking the OK button. If the algorithm succeeds, the Results tab is selected and the lowest-cost result is rendered, as shown on the left side of Figure 3.3. The rendering shows a colored path from beginning to end of the sequence, with the first helix colored blue and subsequent helices colored in blue-green, green, and yellow along the sequence. Sheets are colored dark yellow, orange, and red. Paths between helices and sheets are colored in a gradient so that the complete path has smooth color transitions from beginning to end. Numbers rendered in white near helices and sheets indicate the ordering of secondary structures along the sequence.

The sequence information is displayed in the table on the right side of Figure 3.3. The first column shows the $\alpha$-helices and $\beta$-sheets in the sequence and the second column shows the corresponding structures in the density map. The percentage in parentheses in the second column is the probability of that helix-to-helix or strand-to-sheet pairing occurring in the top 35 results. If the percentage is 100%, this pairing appears in all 35 results returned by the algorithm; if around 50% it appears in roughly half of the results; if close to 0%, it appears in few. We will show in Chapter 4 that pairings with high percentages tend to correspond to the ground truth. Therefore, the user can look for high percentages when trying to select a good result among the 35 results of the algorithm.

At the top of the window is a pull-down menu that allows the user to switch between the lowest-cost correspondence (selected by default) and any of the other correspondences in the top 35, ranked by matching score. When the user selects a different correspondence, the selected correspondence is rendered in the left pane and its sequence information is shown on the right. Figure 3.4 shows a different search result. Comparing to 3.3, note the

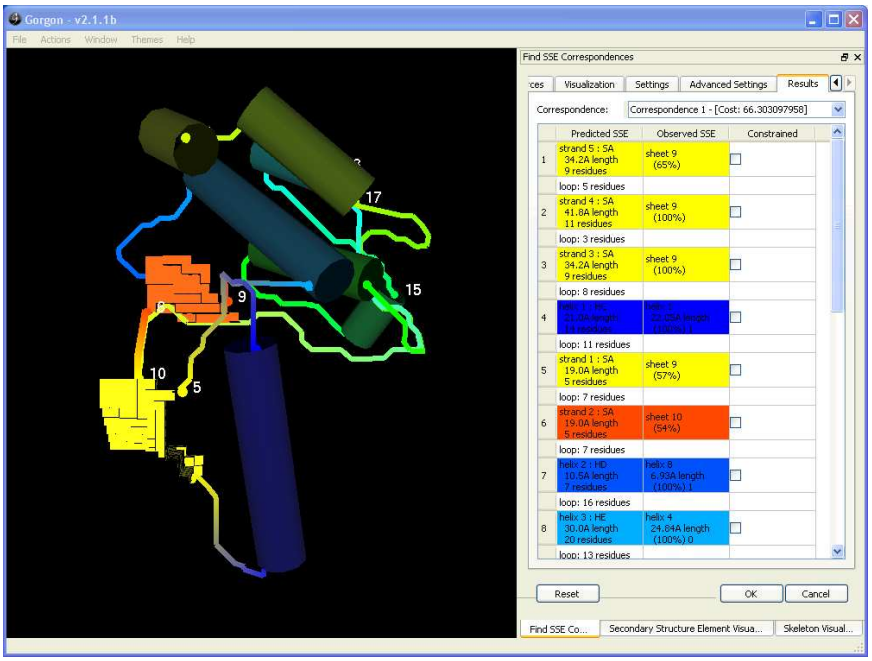



**Figure 3.3:** User interface showing the lowest-cost correspondence results for 1IRK

different path along the sequence, the different coloring of the rendered helices and sheets, and the different list of correspondences in the table on the right.

If the user is satisfied with one of the top 35 results, the work is done. The selected correspondence can be used as an input to further steps of the modeling pipeline in Gorgon.

If the user is partially satsified with one of the top 35 results, it is possible to constrain part of the result and re-run the search to fill in the remaining parts of the correspondence. Section 3.4 describes the process of adding and removing constraints.

If the user is not satisfied with any of the top 35 results, it may be necessary to change some of the algorithm's parameters and re-run the correspondence search. Section 3.5 describes all the parameters.

## 3.4 Adding Constraints

A constraint is a fixed mapping between a secondary structure ($\alpha$-helix or $\beta$-sheet) in the density map and a secondary structure ($\alpha$-helix or $\beta$-strand) in the sequence. Constraints allow the user to include domain knowledge into the search process; for example, if the

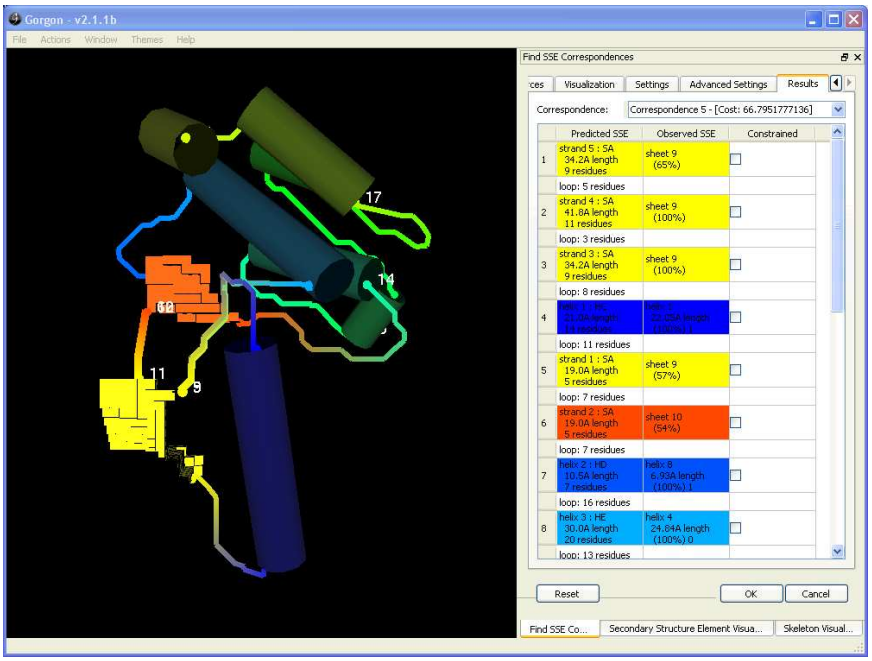
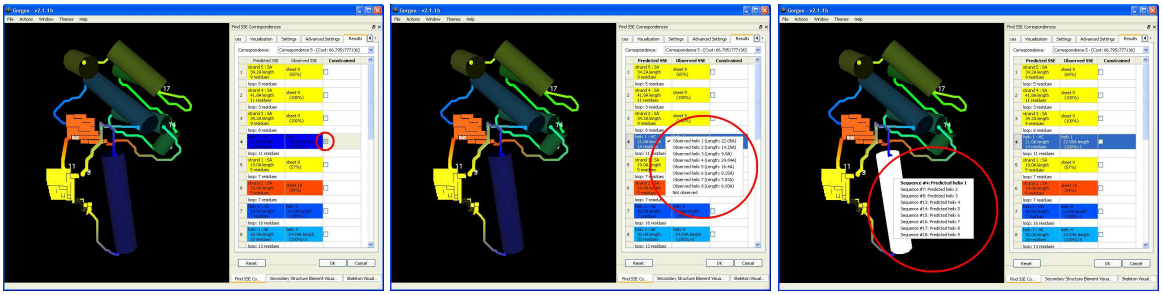



**Figure 3.4:** User interface showing the fifth-lowest-cost correspondence results for 1IRK

user knows the correspondence between part of the sequence and part of the density map, it is possible to constrain just the known part and use the algorithm to find the correspondence for the rest of the sequence.

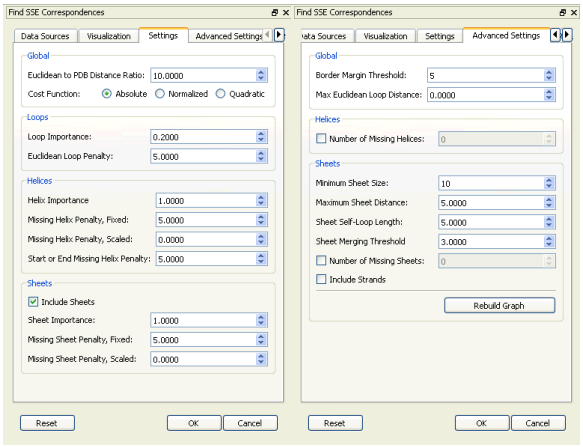
Constraints can also be used to reduce the computational complexity of finding a correspondence. For example, if a structure has many helices and sheets the algorithm may run out of memory, in which case a memory error is reported to the user along with a suggestion to add constraints. In this case the user can use domain knowledge to match a subset of the helices or sheets, specify those matches as constraints, and run the algorithm again to find the best correspondence of the unconstrained structures. Another approach for large input data sets is to first run the algorithm on $\alpha$-helices only, find a good correspondence, constrain all the helices, and then use the algorithm to fill in the $\beta$-sheet correspondences.

The user interface provides three ways to add a constraint. To constrain a matching found by the correspondence algorithm, the user checks the constraint box in the appropriate row in the results list. Another way to add a constraint is to right-click an item in the second row of the results list. This raises a menu that allows the user to select one of the helices in the density map as a constraint. The final way to add a constraint is to right-click a structure in



(a) Check a constraint box in the table (b) Right-click on a helix or strand in the table (c) Right-click on a rendered helix or sheet

**Figure 3.5:** User interface for adding constraints




(a) Settings tab (b) Advanced settings tab

**Figure 3.6:** User interface for changing algorithm parameters

the rendering. This raises a menu that allows the user to select one of the structures in the sequence. The three methods of adding constraints are shown in Figure 3.5.

After adding a constraint, the correspondence algorithm can be run again by clicking the OK button.

## 3.5 Adjusting Parameters

The parameters for the correspondence algorithm are located on two tabs, “Settings” and “Advanced Settings”. Screenshots of these tabs are provided in Figure 3.6, and the parameters are described in Tables 3.3 and 3.4.

| Parameter | Description |
|---|---|
| Euclidean to PDB Distance Ratio | Show or hide the edges in the graph connecting one secondary structure to another. |
| Cost Function | The sizes of helices, loops, and sheets are compared in the cost function using either the absolute difference, the relative difference, or the squared difference. |
| Loop Importance | The relative weight of the loop matching cost in the cost function |
| Euclidean Loop Penalty | The additional penalty incurred when a Euclidean edge is included in a result. Higher penalty discourages the use of Euclidean edges. |
| Helix Importance | The relative weight of the helix matching cost in the cost function. |
| Missing Helix Penalty, Fixed | The fixed penalty incurred when a helix is skipped. |
| Missing Helix Penalty, Scaled | The length-dependent penalty incurred when a helix is skipped. |
| Start or End Missing Helix Penalty | The extra penalty incurred when a helix at the beginning or end of the sequence is skipped. |
| Include Sheets | Determines whether or not sheets are included in the density map graph |
| Sheet Importance | The relative weight of the sheet-strand matching cost in the cost function. |
| Missing Sheet Penalty, Fixed | The fixed penalty incurred when a strand is skipped. |
| Missing Sheet Penalty, Scaled | The length-dependent penalty incurred when a strand is skipped. |

**Table 3.3:** Algorithm parameters available on the Settings tab

| Parameter | Description |
|---|---|
| Border Margin Threshold | Used in building the density map graph. Due to skeleton errors, some loops may not intersect a VRML helix cylinder at the cylinder cap. This threshold allows paths to intersect a helix on the helix side, within some distance of the cap. |
| Max Euclidean Loop Distance | Used in building the density map graph. If two secondary structures are within this distance of each other, a loop is added between these two nodes even if they are not connected along the skeleton. |
| Number of Missing Helices | The number of allowed missing helices in the search. |
| Minimum Sheet Size | Used in building the density map graph. Skeleton surfaces smaller than this threshold are not treated as sheets in the graph. |
| Maximum Sheet Distance | Used in building the density map graph. Skeleton surfaces farther than this distance from VRML sheets are not treated as sheets in the graph. |
| Sheet Self-Loop Length | Used in building the density map graph. A constant representing the length of a loop in the graph from one sheet back to itself. |
| Sheet Merging Threshold | Used in building the density map graph. Skeleton sheets closer together than this distance are treated as a single sheet. |
| Number of Missing Sheets | The number of allowed missing strands in the search. |
| Include Strands | Used in building the graph. Determines whether or not strands are included in the sequence graph. |
| Rebuild Graph | Rebuild both graphs. This is automatically done whenever the user changes a parameter that affects graph construction. |

**Table 3.4:** Algorithm parameters available on the Advanced Settings tab

## 3.6 Automation Using Settings Files

The previous sections describe how to load data, add constraints, and set algorithm parameters using the user interface. Another way to provide the data filenames, constraints, and parameters is by storing them in a text file called a Settings file. A valid Settings file must contain minimally the names of input files and can optionally include constraints and algorithm parameters.

If a user repeatedly runs the same correspondence search on one data set, the use of a Settings file can help a user prevent tedious re-entry of the same filenames and parameters for each search. Settings files can also be saved by the user for later use.

The Settings file is a text file with each row representing a filename, a constraint, or a parameter value.

# Chapter 4

# Results

In this chapter we evaluate the performance of our method on various cryo-EM data sets for which the ground truth structure is known. We provide results showing that the ground truth structure is often returned as one of the top 35 results of our method, and that in many cases it is the top result. We further show that a simple voting scheme based on the top 35 results gives an accurate estimate of the protein structure, even for data sets for which the ground truth is not among the 35 best results. We also present cryo-EM data sets for which our current implementation cannot provide any results due to memory limits, and we describe methods of working around this limitation.

## 4.1 Setup

We present results for eleven cryo-EM volumes at 4Å-10Å resolution, nine of which are simulated from atomic models obtained from the Protein Data Bank [14] and two which are authentic cryo-EM reconstructions (RDV P8 at 6.8Å, GroEL-Apical domain at 4.2Å[1]).

For each structure, our method requires a geometric skeleton (created from a density map), locations and sizes of $\alpha$-helices and $\beta$-sheets on the skeleton, and knowledge of the positions and lengths of $\alpha$-helices and $\beta$-strands in the protein sequence. In our experiments the skeleton is created using the methods of Ju et al. [21] and Abeysinghe et al. [2]. Helices and sheets are found in the density map using SSEHunter [4]. Information about the protein sequence is taken from the Protein Data Bank. Table 4.1 lists the resolution of each input density map, the numbers of helices and sheets in each map, and the numbers of helices and strands in each sequence.

[1]EMDB numbers for these authentic reconstructions are 1060 (RDV P8) and 5001 (GroEL)

| Protein | Volume Size ($d^3$) | Sequence | | Density Map | | |
|---|---|---|---|---|---|---|
| | | Helix count | Strand count | Sheet count | Missing helices | Missing sheets |
| 1UF2 | 96 | 4 | - | - | - | - |
| 2ITG | 64 | 6 | 4 | 1 | - | - |
| 1IRK | 96 | 9 | 9 | 3 | 1 | 1 |
| 1WAB | 64 | 9 | 5 | 1 | 2 | - |
| 1DAI | 64 | 9 | 9 | 3 | - | 2 |
| 1BVP | 128 | 10 | 14 | 3 | - | - |
| Rotavirus | 96 | 14 | 16 | 2 | 5 | 2 |
| 3LCK | 64 | 12 | 7 | 2 | 5 | - |
| 1TIM | 96 | 12 | 8 | 1 | 3 | - |
| GroEL (Apical Domain) | 100 | 5 | 8 | 2 | - | - |
| RDV P8 | 96 | 14 | 8 | 3 | 2 | 1 |

**Table 4.1:** Data used to evaluate our method for finding the correspondence between SSEs.

**A note about algorithm parameters:** The parameters used by our method are detailed in Section 3.5. In all experiments described here, the missing helix and sheet penalty terms in Equation 2.4 are set to $\omega_h = 5, \omega_s = 5$. In our most noisy data set (RDV P8), a Euclidean distance threshold of $\varepsilon = 10\mathring{A}$ was used to create extra edges in the volume graph to allow for missing connectivity in the geometric skeleton.

## 4.2 Evaluation Methods

Because we anticipate that our method does not always find the correct correspondence as its first result, we compute a list of candidate correspondences between the SSEs in the sequence graph and the density map graph, ranked by their matching costs. This can be done easily in the best-first search framework by terminating the search after a number of complete matches (typically 35) have been found.

We evaluate the quality of the 35 results returned by our method by comparing them to a manual labeling of the SSEs in the density volume based on the known atomic structure (for simulated data) or a structural homologue (for authentic data). We use two approaches to compare our method's results to the ground truth, as described in the following sections.

### 4.2.1 Rank of Ground Truth

Intuitively, if our method returns the ground truth as its first result, the method works well. The first evaluation method is to compare the ground truth to each of the results returned

by our method, searching for a result that exactly matches the ground truth. The rank of the correct result is a measure of the quality of the algorithm.

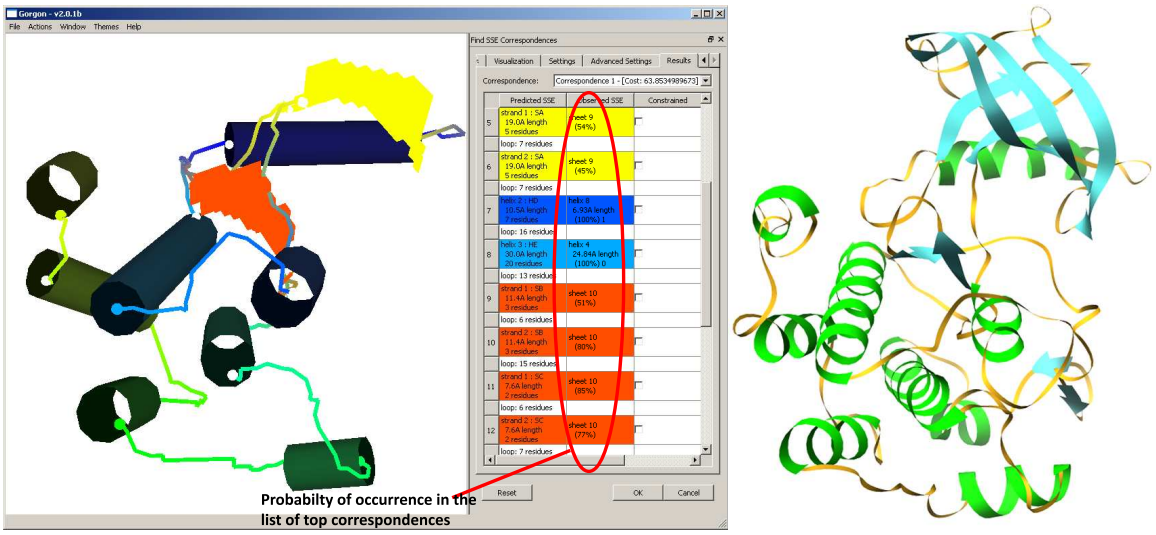
This evaluation method is meaningful only if the ground truth is among the top results of the algorithm. This method does not provide a sense of the overall quality of all the results.

### 4.2.2 Composite of All Results

Intuitively, if our method returns a set of possible correspondences which are very similar to each other, and if these correspondences are similar to the ground truth, our method works well. The second evaluation method uses a simple voting scheme to combine all the results returned by our method into a single composite result. Each SSE-pair in this composite result is compared to each SSE-pair in the ground truth. The quality is measured as the percentage of pairs in the composite result that match the ground truth.

Formally, we denote as $\{i, j\}$ a matching between helix or strand $i$ in the sequence and helix or strand $j$ in the density map. Define probability $P(\{i, j\})$ as the probability that matching $\{i, j\}$ occurs in the set of $n$ results output by the algorithm. If this probability is equal to one, every result contains the matching of $i$ to $j$; if the probability is close to zero, the matching of $i$ to $j$ is not common among the results. After the algorithm is finished and $n$ results have been returned, it is straightforward to compute $P(\{i, j\})$ for all $i$, $j$. As noted in Section 3.3, these probabilities are reported as percentages in the user interface.

Given a set of $n$ results and probabilities $P$, our voting scheme simply chooses for each $i$ the value of $j$ that maximizes $P(\{i, j\})$. This creates a composite result representing the most likely correspondence. We compare each SSE in this composite result to the ground truth and report the fraction of helices that are correct, denoted $E_h$, the fraction of strands that are correct, denoted $E_s$, and the overall fraction of strands and helices that are correct, denoted $E_c$. If $E_h$, $E_s$, and $E_c$ are all equal to one, the voting scheme perfectly predicts the ground truth; if they are close to zero, the voting scheme is not effective at predicting the ground truth.




(a) Results displayed as a pseudo-backbone, and the probabilities for each pairing

(b) Actual $C_\alpha$ backbone

**Figure 4.1:** (a) The user interface showing our method's results for the SSE correspondence of the 1IRK protein of the Human Insulin Receptor, with the pseudo-backbone displayed on the left and the individual SSE correspondences displayed in the table at right along with the probability of each matching in the list of candidates. Even though the correspondence does not have a perfect sheet matching, the pseudo-backbone is almost identical to that of (b) the ground truth.

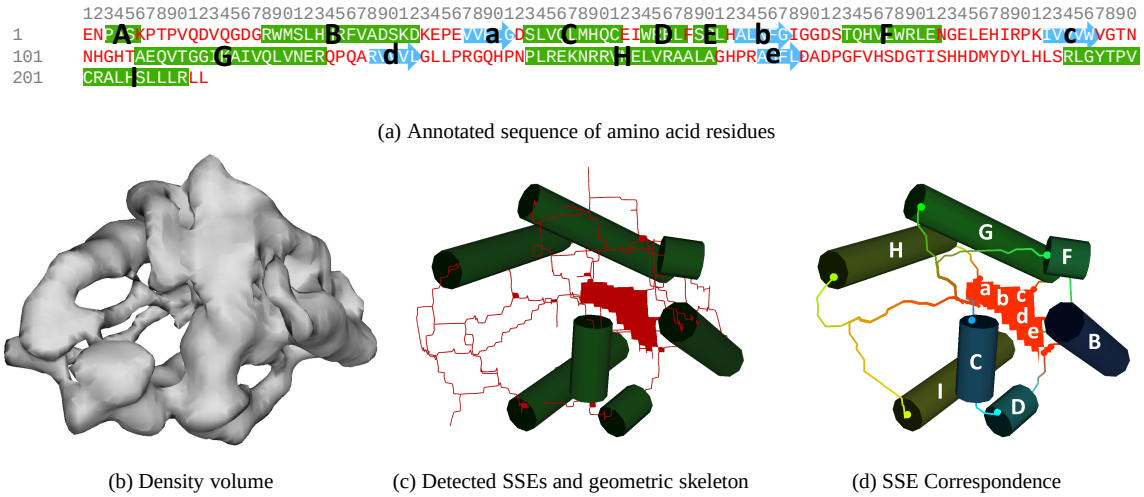
## 4.3 Experiments

We apply our method to the data sets in Table 4.1 and evaluate the results using the two methods described above: we compute the rank of the ground truth among the candidate matches, and we compute $E_h$, $E_s$, and $E_c$. In our experiments we observe that an $E_h$, $E_s$, or $E_c$ score higher than 0.8 indicates a very good list of candidate matches that differ from the ground truth by only one or two SSEs. We report these results in Table 4.2. Details about several of these data sets are provided in the sections that follow.

### 4.3.1 Unsupervised Helix and Sheet Matching

#### 1IRK

Figure 4.1 shows the result of applying our method to find the SSE correspondence for the 1IRK protein. This data set is challenging due to missing SSE elements and similar lengths of loops, strands and helices. Note that the pseudo-backbone generated for the first candidate is almost identical to the ground truth backbone, in spite of the fact that two




(a) Annotated sequence of amino acid residues




(b) Density volume (c) Detected SSEs and geometric skeleton (d) SSE Correspondence

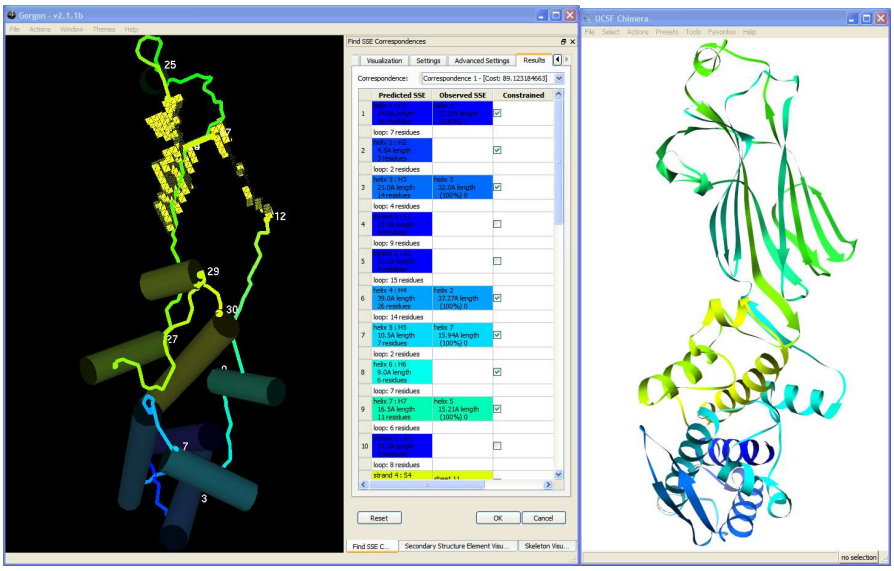
**Figure 4.2:** (a) The annotated sequence of amino acid residues of the 1WAB protein. (b) The density volume. (c) The detected secondary structure elements and the skeleton. (d) The correspondence between the two sets of SSEs computed by our method.

$\beta$-strands were not correctly identified by our method as missing in the density map. The helix portion of the pseudo-backbone exactly matches the ground truth, so the helix-only rank in Table 4.2 is 1. By contrast, the ground truth strand correspondence does not appear in any of the top 35 results, so the helices-and-sheets rank in the table is $>35$.

The metrics $E_h$, $E_s$, $E_t$ for these 35 results are 1.0, 0.76, and 0.92, respectively. This indicates that if we take a majority vote among the top 35 correspondences, we would obtain the correct helix correspondence and more than three fourths of the correct sheet correspondences. This means that the user can use the probabilities $P(\{i,j\})$ associated with each SSE to select a good result from among the top 35, and to make assumptions while iteratively refining the results. These probabilities are reported in the user interface as percentages, as shown by the red circle in Figure 4.1.

### 1WAB

Figure 4.2 shows that our method is able to identify the correct SSE assignment for 1WAB as a highly-ranked candidate. Observe that our algorithm is robust to noise in the data, such as the two missing helices in the density volume. As a by-product of our algorithm, a pseudo-backbone can be visualized by rendering the skeleton paths represented by the graph edges in the optimally matching chain. This pseudo-backbone serves as a starting point when determining the actual $C_\alpha$ backbone.



(a) Rank 1 correspondence (b) Actual correspondence

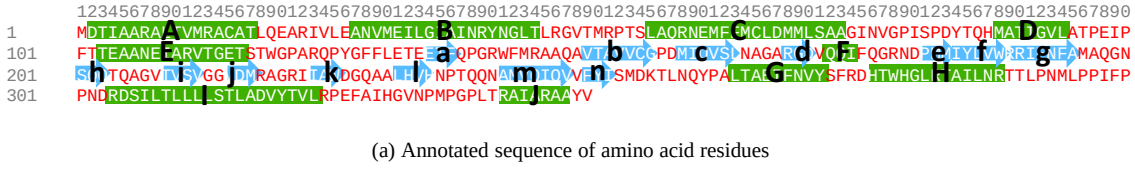
**Figure 4.3:** (a) The optimal correspondence for Rotavirus computed by our method. (b) The actual correspondence.

### 4.3.2 Unsupervised Helix-Only Matching

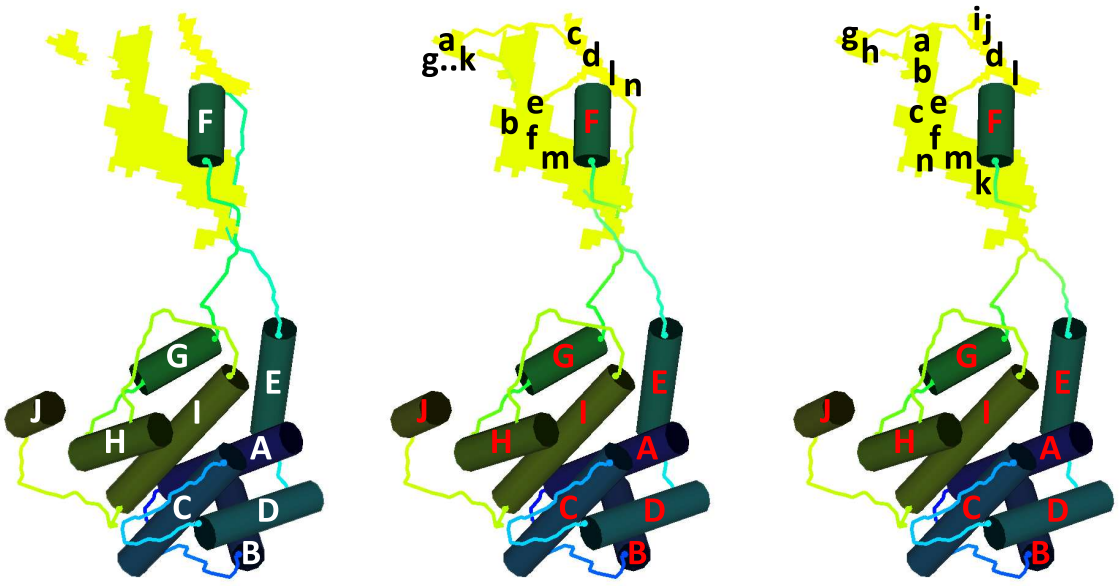
As the number of $\alpha$-helices and $\beta$-sheets in a protein increases, the memory required by our method increases exponentially in the worst case. We observe that our implementation runs out of memory when the number of SSEs reaches 15 to 30, depending on the connectity of the skeleton. For proteins larger than this limit, we use one of two approaches to work around the memory limitation.

The first approach, described in this section, is to use a two-step process. In the first step all $\beta$-sheets are removed from the graph and a good $\alpha$-helix-only correspondence is found. The helix correspondences are fixed according to the result of this first step, and our algorithm is used a second time solve for only the $\beta$-sheet correspondences. We find that this approach works best for proteins where all $\beta$-sheets are located in a single cluster and not surrounded by $\alpha$-helices.

The second approach, adding constraints based on the user's knowledge of protein structure, is described in Section 4.3.3.



(a) Annotated sequence of amino acid residues



(b) Helix-only correspondence (c) Optimal correspondence after helix constraints (d) Actual correspondence

**Figure 4.4:** (a) The annotated sequence of amino acid residues of the 1BVP protein. (b) The optimal correspondence computed by our method where the helix correspondence is correct. (c) The optimal correspondence after the helices have been constrained. (d) The actual correspondence.

## Rotavirus

For the Rotavirus structure shown in Figure 4.3, we first solved for the $\alpha$-helix-only correspondence. As shown in Table 4.2, our method returns the actual $\alpha$-helix correspondence as the lowest-cost match. Next we constrain all $\alpha$-helices and solve for the $\beta$-strand correspondence. Due to the clustered nature of the $\beta$-strands, the correct $\beta$-strand correspondence does not appear in the top list of candidates. The relatively poor sheet matching results are reflected by the low value of $E_s$.

**1BVP**

We apply the same strategy to find the structure of the 1BVP protein of the Blue-Tongue Virus, with results shown in Figure 4.4. Like the Rotavirus example, our algorithm correctly predicts the helix correspondence but returns relatively poor sheet matching results, as demonstrated by the low value of $E_s$ in the results table. The cause of the poor sheet matching results is that two long $\beta$-strands are incorrectly matched to a very large $\beta$-sheet, when they should be matched to a separate, smaller sheet. This error is due to our algorithm's assumption that longer strands should be matched to larger sheets.

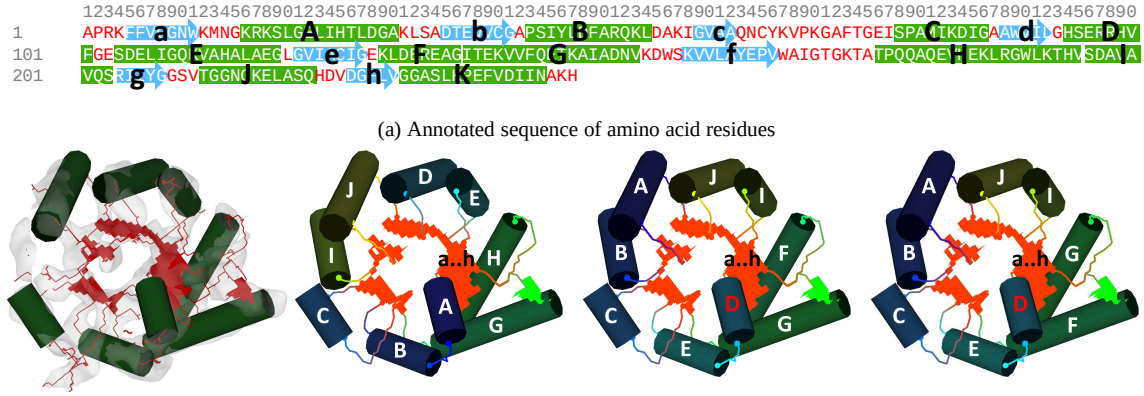
### 4.3.3 Interactive Matching

The previous section described a two-step approach for working around memory limitations for structures with many SSEs where strands are clustered together and separate from helices. This is just one possible cause of memory problems. Other possible causes include a high degree of symmetry, such as a $\beta$-barrel structure, or a large number of $\alpha$-helices.

Even if the algorithm is not limited by memory, some types of input data, such as a low-resolution density map with insufficient shape or topology information about the protein, can lead to incorrect matching results. To overcome these limitations, we allow the user to manually assign matching constraints based on knowledge about protein structure.

Specifically, the user can designate the correspondence between a subset of helices and/or sheets in the sequence graph and the density map graph. This information is translated into node attributes in the graph, reducing the branching factor of the search. The user interface for adding constraints is described in Section 3.4.

Although the process of choosing constraints can be time consuming, we note that it is significantly faster than finding a complete correspondence manually. Our implementation enables relatively fast iteration as constraints are added and removed, with typical algorithm execution times ranging from a few seconds to forty seconds on a desktop computer, as shown in Table 4.2.



(a) Annotated sequence of amino acid residues

(b) Density, skeleton and SSEs (c) Optimal correspondence (d) Optimal after constraint (e) Actual correspondence

**Figure 4.5:** (a) The annotated sequence of amino acid residues of the 1TIM protein. (b) The inputs to our algortihm. (c) The minimum-cost correspondence computed by our method. (d) The minimum-cost correspondence computed by our method, after the user has constrained the helix labeled D. (e) The actual correspondence, which is returned by our method as the 25th result after the constraint in (d) has been applied.

### 1TIM

Figure 4.5 shows an example of the 1TIM protein found in chicken muscle. For this structure our implementation runs out of memory if no user constraints are applied, even if sheets are removed from the problem. After a user-specified constraint was added to the helix marked as (D), the correct correspondence was found as the 25th result in the candidate list. Although the ground truth has rank 25, the large values of $E_h$, $E_s$, and $E_t$ in Table 4.2 show that the composite accuracy of the top 35 correspondence results is quite good.

In this example, the constraint was chosen by noting that in the sequence, most paths from one helix to another pass through a sheet, with only one exception. The case where two helices are connected only by a loop should correspond to the two helices in the density that are connected by a loop.

## 4.4 Summary of Results

The results for all 11 proteins are presented in Table 4.2, showing the number of user-specified constraints used, the rank of the ground truth when considering only the helix correspondences, the rank of the ground truth considering all of the SSEs, the execution times, and the associated $E_h$, $E_s$, and $E_c$ scores for the list of top 35 candidates.

| Protein | User Constraints | Helix only | | Helices and Strands | | | Execution time (s) | |
|---|---|---|---|---|---|---|---|---|
| | | Rank | $E_h$ | Rank | $E_s$(Strand only) | $E_c$(Composite) | First match | Top 35 matches |
| 1UF2 | - | 1 | 0.96 | 1 | - | 0.96 | 0.00 | 0.00 |
| 2ITG | - | 1 | 1.00 | 1 | 1.00 | 1.00 | 0.00 | 0.01 |
| 1IRK | - | 1 | 1.00 | >35 | 0.76 | 0.92 | 0.65 | 0.82 |
| 1WAB | - | 11 | 0.89 | 11 | 1.00 | 0.91 | 0.65 | 1.14 |
| 1DAI | 1 | 17 | 0.82 | 17 | 1.00 | 0.89 | 0.21 | 0.32 |
| 1BVP | -/10* | 1 | 1.00 | >35 | 0.58 | 0.83 | 0.86 | 1.50 |
| Rotavirus | -/14* | 1 | 1.00 | >35 | 0.19 | 0.62 | 5.25 | 6.83 |
| 3LCK | - | 7 | 0.95 | >35 | 0.71 | 0.89 | 17.01 | 21.04 |
| 1TIM | 1 | 25 | 0.91 | 25 | 1.00 | 0.93 | 11.60 | 15.41 |
| Groel | - | 1 | 1.00 | >35 | 0.90 | 0.95 | 0.07 | 0.11 |
| RDV P8 | 6 | 12 | 0.96 | >35 | 0.88 | 0.94 | 36.54 | 38.32 |

**Table 4.2:** The number of user constraints specified for each experiment, the rank and $E_h$ score when considering only the $\alpha$-helix correspondence, the rank and $E_s$ scores when considering both $\alpha$-helix as well as $\beta$-strand correspondences, the combined $E_c$ scores, and the execution times to find the first correspondence and the list of the top 35 candidates. *For 1BVP and Rotavirus, the experiment was conducted by first finding the helix correspondence using no constraints, and then constraining all the helices to find the $\beta$-strand correspondence.

We have presented three different ways to apply our method: solve for helix and sheet correspondences together in an unsupervised way; solve first for helix correspondences, then solve for sheet correspondences as a second step; or solve for the correspondence in an interactive way by incorporating domain knowledge into the algorithm's cost function.

The results in Table 4.2 show that these three approaches can be used to find the correct $\alpha$-helix correspondence, and to a lesser extent, to find the correct $\beta$-sheet correspondence, for several proteins of different sizes and shapes. The good quality of the results is indicated by the low rank of the ground truth in the results, and by the high values of $E_h$ for helices, $E_s$ for strands, and $E_c$ for the overall correspondence.

## 4.5 Performance

The execution times listed in Table 4.2 show that our implementation is fast enough to be used interactively, allowing the user to run the algorithm many times, adding and removing constraints or adjusting cost function parameters between subsequent iterations. All tests were performed on a desktop computer running 32-bit Windows XP.

We noted above that our method requires more memory as the number of SSEs increases or as the connectivity of the skeleton increases. We have described some ways to work around this limitation in order to build models of increasingly large proteins.

# Chapter 5

# Discussion

We have described a method of finding the correspondence between the secondary structures in a protein sequence and the secondary structures in an intermediate-resolution protein density map. Our method builds on previous work from our group which considered only $\alpha$-helices [1]; our main contribution is the inclusion of $\beta$-sheets into the method.

We have implemented our method in C++ using Phython and Qt for the user interface and visualization, and we have described the features of our user interface, which is part of the Gorgon protein modeling system.

We have presented results showing that our method successfully finds $\alpha$-helix and $\beta$-sheet correspondence on a collection of real and simulated data sets. We achieve better results for helices than sheets. We showed that in some structures such as the $\beta$-barrel, the inclusion of sheets in the algorithm improves the quality of the helix results.

## 5.1 Limitations

**Memory** Due to memory limitations, our method cannot automatically solve for correspondences in structures with a large number of helices and sheets. In Chapter 4 we presented two approaches for working around this issue. The first approach is a two-step process in which the helix correspondence is found, then the helix correspondence is fixed and the correspondence for the sheets is found. The second approach is to use use our algorithm in a semi-interactive fashion, with the user specifying portions of the correspondence and the algorithm solving for the rest.

**Skeleton Quality** Our algorithm relies on a geometric skeleton to determine possible paths between SSEs, which means that the results generated by our method are highly depenedent on the quality of the skeleton. For skeletons with many broken paths, we suggest that the user refine the skeleton using the Grayscale Skeletonization method included in the Gorgon system.

**$\beta$-Sheet Marching Accuracy** Our results showed that the $\beta$-sheet matching accuracy of the algorithm is inferior to the $\alpha$-helix matching accuracy. Improving the algorithm design may be able to improve this to some extent. However some of this difference is due to the input to our algorithm: it is more difficult to accurately locate sheets than helices in a density map.

## 5.2 Future Work

**$\beta$-Sheets and $\beta$-Strands** In the cost function, each match between a sheet in the density map and a strand in the sequence considers only the average expected size of the sheet, and does not include any information about the geometry of the sheet or the points at which the path enters and exits the sheet. In proteins, strands tend to form parallel paths across sheets with entry and exit points on opposite sides of a sheet. The cost function could be modified to discourage non-parallel paths through sheets.

**Loop Lengths** We determine the lengths of loops by measuring the shortest path between two structures across the skeleton. The true path between two nodes may follow a much longer path, particularly if the skeleton is dense with connections between structures. This suggests that our method might be improved by associating with each pair of structures a range of possible path lengths, and modifying the cost function so that the matching cost is low as long as the path lengths fall within this range.

**Constraints** The current implementation allows the user to specify constraints between structures in the density map and the sequence. In addition to this, neighbor constraints could be quite powerful. The user could specify that two structures in the density map should be matched to two neighboring structures along the sequence.

# References


[1] Sasakthi Abeysinghe, Tao Ju, Matthew L. Baker, and Wah Chiu. Shape modeling and matching in identifying 3D protein structures. *Computer-Aided Design*, 40(6):708–720, June 2008.

[2] Sasakthi S. Abeysinghe, Matthew Baker, Wah Chiu, and Tao Ju. Segmentation-free skeletonization of grayscale volumes for shape understanding. In *Proc. IEEE International Conference on Shape Modeling and Applications SMI 2008*, pages 63–71, 2008.

[3] Matthew L. Baker, Sasakthi S. Abeysinghe, Stephen Schuh, Ross A. Coleman, Austin Abrams, Michael P. Marsh, Corey F. Hryc, Troy Ruths, Wah Chiu, and Tao Ju. Modeling protein structure at near atomic resolutions with gorgon. *Journal of Structural Biology*, 174(2):360 – 373, 2011.

[4] Matthew L. Baker, Tao Ju, and Wah Chiu. Identification of secondary structure elements in intermediate-resolution density maps. *Structure*, 15(1):7–19, January 2007.

[5] H. Bunke. Error correcting graph matching: On the influence of the underlying cost function. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(9):917–922, 1999.

[6] H. Bunke and G. Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1:245–253, 1983.

[7] Horst Bunke and Bruno T. Messmer. Recent advances in graph matching. *IJPRAI*, 11(1):169–203, 1997.

[8] J. Cheng, A. Z. Randall, M. J. Sweredoski, and P. Baldi. Scratch: a protein structure and structural feature prediction server. *Nucleic Acids Research*, 33:W72–W76, 2005.

[9] Wah Chiu, Matthew L. Baker, Wen Jiang, Matthew Dougherty, and Michael F. Schmid. Electron cryomicroscopy of biological machines at subnanometer resolution. *Structure*, 13(3):363–372, March 2005.

[10] William J. Christmas, Josef Kittler, and Maria Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):749–764, 1995.

[11] Christian Cole, Jonathan D. Barber, and Geoffrey J. Barton. The jpred 3 secondary structure prediction server. *Nucleic Acids Research*, 36:W197–W201, May 2008.

[12] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.

[13] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. Performance evaluation of the vf graph matching algorithm. In *International Conference on Image Analysis and Processing*, 1999.

[14] S. Dutta and H.M. Berman. Large macromolecular complexes in the protein data bank: a status report. *Structure*, 13:381–388, 2005.

[15] J. Feng, M. Laumy, and M. Dhome. Inexact matching using neural networks. *Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies, and Hybrid Systems*, pages 177–184, 1994.

[16] Research Collaboratory for Structural Bioinformatics. RCSB Protein Data Bank Statistics. `http://www.rcsb.org/pdb/static.do?p=general_information/pdb_statistics/index.html`, 2011. [Online; accessed 18-Apr-2011].

[17] L. Herault, R. Horaud, F. Veillon, and J. J. Niez. Symbolic image matching by simulated annealing. In *Proc. British Machine Vision Conference (BMVC90)*, pages 319–324, 1990.

[18] Radu Horaud and Thomas Skordas. Stereo correspondence through feature grouping and maximal cliques. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(11):1168–1180, 1989.

[19] Wen Jiang, Matthew L. Baker, Steven J. Ludtke, and Wah Chiu. Bridging the information gap: computational tools for intermediate resolution structure interpretation. *Journal of Molecular Biology*, 308(5):1033 – 1044, 2001.

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

[21] Tao Ju, Matthew L. Baker, and Wah Chiu. Computing a family of skeletons of volumetric models for shape description. *Computer-Aided Design*, 39(5):352–360, 2007.

[22] Yifei Kong and Jianpeng Ma. A structural-informatics approach for mining [beta]-sheets: Locating sheets in intermediate-resolution density maps. *Journal of Molecular Biology*, 332(2):399 – 413, September 2003.

[23] Yifei Kong, Xing Zhang, Timothy S. Baker, and Jianpeng Ma. A structural-informatics approach for tracing [beta]-sheets: Building pseudo-c[alpha] traces for [beta]-strands in intermediate-resolution density maps. *Journal of Molecular Biology*, 339(1):117 – 130, May 2004.

[24] B. T. Messmer and H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):493–504, 1998.

[25] N.J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann Publishers, 1980.

[26] L. Roberts, R. J. Davenport, E. Pennisi, and E. Marshall. A history of the human genome project. *Science*, 291(5507):1195, February 2001.

[27] A. Sanfeliu and K.S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. Systems, Man, and Cybernetics*, 13:353–363, 1983.

[28] L. G. Shapiro and R. M. Haralick. Structural descriptions and inexact matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 3(5):504–519, 1981.

[29] W. H. Tsai and K. S. Fu. Error-correcting isomorphisms of attributed relational graphs for pattern recognition. *IEEE Trans. Systems, Man, and Cybernetics*, 9:757–768, 1979.

[30] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, 1976.

[31] Yuan-Kai Wang, Kuo-Chin Fan, and Jorng-Tzong Horng. Genetic-based search for error-correcting graph isomorphism. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 27(4):588–597, 1997.

[32] A.K. Wong, M. You, and A.C. Chan. An algorithm for graph optimal monomorphism. *IEEE Trans. Systems, Man, and Cybernetics*, 20(3):628–636, 1990.

[33] Yinghao Wu, Mingzhi Chen, Mingyang Lu, Qinghua Wang, and Jianpeng Ma. Determining protein topology from skeletons of secondary structures. *Journal of Molecular Biology*, 350(3):571–586, 2005.

[34] Z. Hong Zhou. Towards atomic resolution structural determination by single-particle cryo-electron microscopy. *Curr Opin Struct Biol*, 18(2):218–228, 2008.

[35] Z. Hong Zhou, Matthew Dougherty, Joanita Jakana, Jing He, Frazer J. Rixon, and Wah Chiu. Seeing the herpesvirus capsid at 8.5å. *Science*, 288(5467):877–880, May 2000.

# Vita

Stephen Schuh

**Degrees**

B.A. Physics, Reed College, May 2001
M.S. Computer Science, Washington University, May 2011

**Publications**

Matthew L. Baker, Sasakthi S. Abeysinghe, Stephen Schuh, Ross A. Coleman, Austin Abrams, Michael P. Marsh, Corey F. Hryc, Troy Ruths, Wah Chiu, and Tao Ju. Modeling protein structure at near atomic resolutions with Gorgon. *Journal of Structural Biology*, 174(2):360-373, 2011.

Nathan Jacobs, Stephen Schuh, Robert Pless, Compressive Sensing and Differential Image Motion Estimation, *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.

May 2011

**Protein Structure from Cryo-EM Data, Schuh, M.S. 2011**