Reconstruction of Protein Structure Similarity Using Contact Map Overlap Measure

Critique

Satabdi Aditya

Committee:

Dr. Robert Sloan (Chair)
Dr. Bhaskar DasGupta
Dr. Mark Grechanik

Department of Computer Science
University of Illinois at Chicago

March 2014

Abstract

A fundamental problem in molecular biology, with applications in domains like medicine, biotechnology, computational biology and chemistry, is to study the three-dimensional structure of a protein as the functional characteristics of a protein are attributed to its structure. Proteins that are similar in structure tend to share similar properties; hence finding similarity measure between two given proteins has garnered sufficient interest. Out of the several measures proposed, Contact Map Overlap (CMO) is one of the most reliable and robust measures of protein structure similarity. The measure is maximized when the amino acid residues of two proteins are aligned to maximize the number of common residue contacts. Additionally, such measure facilitates the grouping of proteins into functionally similar clusters and can potentially incorporate experts' feedback in the modeling process. Quite justifiably, significant research effort has been invested over the last two decades in the maximum CMO problem and this report, in its limited page budget, presents a brief summary of some of the most important works that have shaped the course of the research in this problem.

To begin with, the report provides a formal description of the maximum CMO problem and illustrates that the calculation of the overlap of contact maps between two proteins is an NP-hard problem. Then the approximation algorithms for both 2 and 3-dimensional structures are explored which is followed by a succinct description of an exact algorithm based on integer programming that provably performs better than other existing algorithms for maximum CMO problem in 2-dimensional structure. Finally, a critical analysis of the algorithms discussed is appended and possible extensions and research directions are previewed.

Contents

Chapter 1

Introduction

Since proteins are the base of a variety of roles such as acting as enzymes, transporting small molecules, forming cellular structures, regulating cell processes, carrying signals etc, understanding proteins would help in solving a number of unanswered questions which still elude biologists. A protein's primary structure consists of a linear combination of monomers (amino acids) of 20 different kinds. Under various physiological conditions, this linear arrangement folds and adopts a specific geometric pattern called the tertiary structure of the protein. Therefore, residues which are distant in the linear combination can come close to each other in this 3D structure forming a close bond (contact). It is this 3D structure of a protein that determines its macroscopic properties, behaviors and functions. Proteins that are similar in their structures will likely have similarity in functions. Quite naturally, studying the 3D structure of a protein has been an active area of research for the last two decades. This report looks into studying protein structure similarity by aligning a protein to the known structure of another protein. Before going into the details of *Protein Structure Similarity*, in the next two sections, some of the basic definitions are introduced that will be used through the rest of the report.

1.1 Protein Structure Similarity

The 3D structure of the protein has earlier been predicted from its amino acid sequence *via* numerous experimental processes like X-ray crystallography (Ilari and Savino, 2008) or Nuclear Magnetic Resonance Waudby et al. (2013) but efforts have also been made for designing computer algorithms to predict the same. This process of predicting the 3D structure of a protein from its amino acid sequence is called *Protein Folding Problem* (Berger and Leighton, 1998; Crescenzi et al., 1998; Dill and MacCallum, 2012). However, even after significant research, the protein folding problem stays computationally intractable. An alternative approach to study the functional properties of a protein, perhaps computationally more tractable, is to find similarity of a given protein with other proteins. There could be two different types of similarity that one can study – sequence similarity and structural similarity. Sequence similarity is a far less reliable method than structural similarity as it is the structure of the protein that determines its functional properties. To compute the similarity between two protein 3D structures, one needs some scoring mechanism that captures the biological relevance of the chemical and physical constraints involved in molecular recognition. Few such known and available structural similarity measures are listed below:

- *Root Mean Square Division* (RMSD) – measures root of sum of the squares of Euclidean distance between mapped residues (Cohen and Sternberg, 1980; Diamond, 1988, 1992),
- difference between the distance matrices (Holm and Sander, 1993, 1996; Hu et al., 2004),
- *Contact Map Overlap* (CMO)– measures the pairwise distance between residues of each protein (Goldman et al., 1999; Agarwal et al., 2007; Andonov et al., 2011),

• *ad-hoc* score based on local secondary structure, hydrogen bonding pattern or interaction environment (Blundell and Johnson, 1993; Vriend and Sander, 1991; Taylor and Orengo, 1989; Zuker and Somorjai, 1989).

This report deals only with the *Contact Map Overlap* scoring scheme. Of the various structures used to represent a protein, a simplified lattice model is adopted to represent the backbone of the protein to study CMO. The protein is represented as a non-self intersecting path on an integer grid in two or three dimensions, formally known as a self-avoiding walk. Details of such construction and assumption are described in the following sections.

1.2 Contact and Contact Map Overlap (CMO)

Since proteins are not just a collection of arbitrary edges, the structure of a protein is represented on the lattice using a *self-avoiding walk*. As the term suggests, a self-avoiding walk is a sequence of moves on the lattice that will not visit the same point twice, *i.e.* a path from a point to another will never intersect itself. A *contact* helps us measure the closeness of two amino acids which are close in their protein structure forming some bond. When a protein folds to form its 3D structure, various amino acids which are not close in the linear sequence come close to one another forming a bond. The closeness is mostly measured between the centers of mass of amino acid backbone atoms (C_α) or side chain (C_β) atoms. Two such amino acids are said to be in contact if the distance between them is less than a threshold value, typically considered a few angstroms.

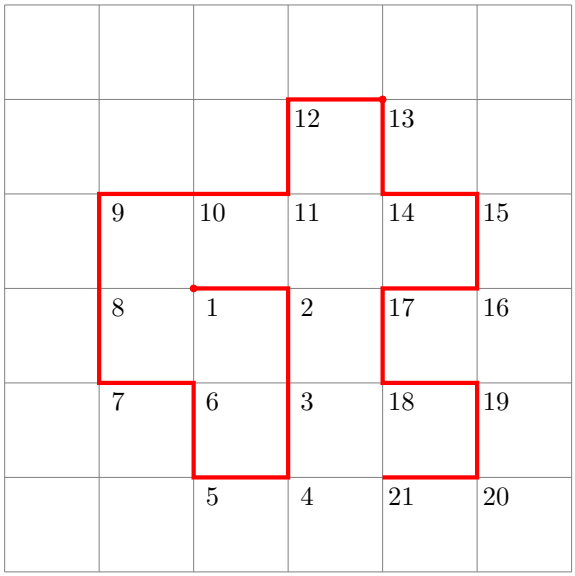

Figure 1.1: A self-avoiding walk

Fig. 1.1 show a self-avoiding walk in a two dimensional lattice which is a non-self intersecting path on the grid. In the self-avoiding walk, the red line shows the path of the walk. In a two-dimensional lattice, each vertex in a self-avoiding walk can have maximum degree two (except the first and the last vertex which might have three). Similarly in a three-dimensional lattice, each vertex can have a maximum degree of four (the first and the last vertex can have five).

A *contact map* is a useful graphical representation(and two-dimensional description) of the structure of a protein where the vertex set represents the amino acids in the same order in which they appear in the linear sequence of the protein and the edge set represents the set of contacts. In the contact map of a self-avoiding

walk, the edges are between vertices which are at a unity distance from each other in the grid but not in the walk. Fig. 1.2 shows the contact map of the self avoiding walk shown in Fig. 1.1. Contact maps of proteins can be used for a wide range of functionality. Few of them are secondary structure prediction, fold identification, fold classification, fold assignment, protein structure alignment and threading (Zhang, 2008). The three-dimensional structure of RNA can also be cast in the form of a contact map and hence this helps in understanding their structure (Zhang, 2008).

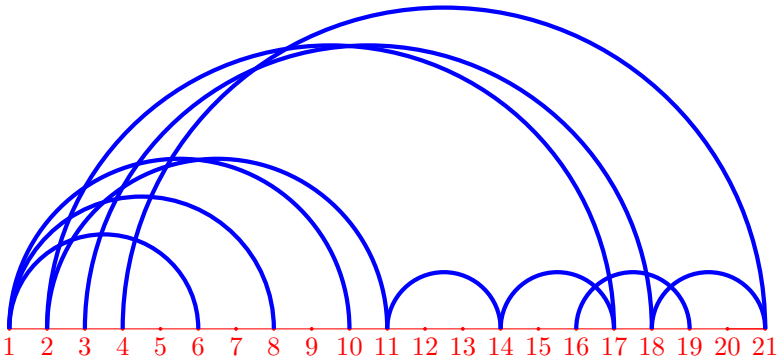

Figure 1.2: A contact map example

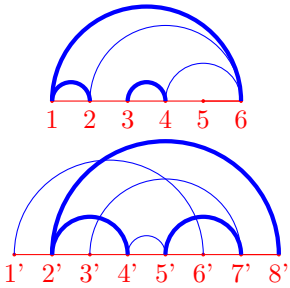

Figure 1.3: A contact map overlap example

Given two contact maps, contact map overlap is defined as the maximum number of contacts preserved by some order-preserving bijection between subsets of the two vertex sets. Let us explain the same with the two contact maps shown in Fig. 1.3. The contact overlap between the two graphs is 3 and these edges are preserved by the order preserving bijection between the two subset of edges $S_1 = \{1, 2, 3, 4, 6\}$ of the first graph and $S_2 = \{2', 4', 5', 7', 8'\}$. Using these fundamental concepts, the next section gives a detailed description of the course in which the research has progressed using contact map overlap measure.

The rest of the report is organized as follows. Chapter 2 presents the relevant literature on approximation algorithms for solving the CMO problem. Chapter 3 briefly describes some of the exact algorithms proposed for solving the CMO problem. Finally, Chapter 4 provides a critical analysis of the algorithms proposed in Chapter 2 and Chapter 3 and discusses about possible extensions and research directions.

Chapter 2

Approximation Algorithms for Maximum CMO

This chapter reviews the approximation algorithms for solving the maximum CMO problem. To begin with, Goldman et al. (1999) represent the problem as a graph theoretic abstraction and prove the NP-hardness of the problem. The authors further propose a special class of CMO problem for which 3-approximation algorithm runs in $O(n^6)$ time. In the next few sections, the 6-approximation algorithm with better time complexity for the same class of problems, as proposed in Agarwal et al. (2007), is discussed. Moreover, an $O(\sqrt{n})$ approximation algorithm, also suggested by Agarwal et al. (2007), for the 3-dimensional self-avoiding walk is summarized. Such results are the first of their kind for 3-dimensional self-avoiding walk.

2.1 CMO as a Graph Theoretic Optimization Problem

Let Z be an integer grid in $\mathbb{R}^2$ or $\mathbb{R}^3$. A *self-avoiding walk* is a one-to-one mapping $f : V = \{v_1, v_2, ..., v_n\} \rightarrow Z$ such that $\|v_i - v_{(i+1)}\| = 1$ and $v_i \neq v_j$, $1 \leq (i, j) \leq n$. Self-avoiding walks are used to represent protein structures as physical evidences show that proteins cannot be represented by arbitrary edges.

A *contact map* is an ordered graph $G = (V, E)$ where $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices and $E = \{(v_i, v_j) : \|j - i\| > 1$ and $\|v_i - v_j\| = 1\}$ is the set of edges. Contact maps are useful representation of proteins where each vertex represents one amino acid and the edges are the proxy for the pairs of amino acids (not in the linear sequence) whose centroids are closer than a fixed threshold. Fig. 1.1 and 1.2 show the examples of a self-avoiding walk and its corresponding contact map.

Given two contact maps $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ where $V_1 = \{v_1, v_2, \cdots, v_{n_1}\}$ and $V_2 = \{u_1, u_2, \cdots, u_{n_2}\}$, n_1 and n_2 being the total number of vertices in V_1 and V_2 and two subsets $S_1 \subseteq \{v_1, v_2, \cdots, v_{n_1}\}$ and $S_2 \subseteq \{u_1, u_2, \cdots, u_{n_2}\}$ with $|S_1| = |S_2|$, the *contact map overlap* is defined as the maximum cardinality of the set $|\{(v_i, v_j) \in E_1 : v_i, v_j \in S_1, (f(v_i), f(v_j)) \in E_2|$ under some *order-preserving bijection* f. Fig. 1.3 explains a contact map overlap between the two graphs.

2.2 NP-Hardness of CMO

The contact map overlap problem, in its graph theoretical representation, is NP-hard. This implies that it is less probable that a polynomial time algorithm exists that computes the contact map overlap between two graphs of self-avoiding walks.

The structure of the proof is suggested in Goldman et al. (1999). The proof of the above is deduced with the help of two other lemmata. In the first step, it is proved that computation of the overlap of two 2-stack graphs which have a maximum degree 3 is NP-hard. The reduction is from the Planar 3-SAT problem which is known to be NP-complete itself (Lichtenstein, 1982). Let the Planar 3-SAT problem instance consist of

n variables $\{x_1, x_2, \cdots, x_n\}$ and m clauses $\{c_1, c_2, \cdots, c_m\}$. This instance of the problem is mapped into two 2-stack graphs G_1 and G_2. These two graphs contain variable and clause gadgets arranged in a linear layout according to a planar embedding of the formula. Between each variable and clause gadgets (nodes in the graph), enforcing gadgets are inserted which consist of $5m$ (on $10m$ nodes) edges. Once the two graphs are constructed, the constraints of the Planar 3-SAT formula is satisfied by an order preserving bijection between the vertices of G_1 and G_2. The order-preserving bijection is as follows. The variable node in each variable gadget of $G_1$is mapped to the true literal node in the corresponding variable gadget of G_2. Further, nodes in the enforcing gadgets of G_1 are mapped to nodes in the corresponding gadget of G_2. Finally, the clause node in each clause gadget of G_1 is mapped to a true literal node in the corresponding clause gadget of G_2. The two graphs can thus be reduced to an instance of the Planar 3-SAT problem and since the Planar 3-SAT problem is NP-complete, it is NP-hard to compute the overlap of the two 2-stack graphs of maximum degree 3.

The second step involves proving that it is also NP-hard to compute the overlap of two 2-stack graphs which have a maximum degree 1. The proof follows similar argument, *i.e.* the problem again is reduced to a Planar 3-SAT problem in polynomial time. The only difference is the way in which the two graphs are constructed such that they are easily decomposable into two stacks each having degree 1. The order-preserving bijection follows the same rules.

These two proofs are successfully used to construct the proof of NP-hardness of the computation of the contact overlap of two self-avoiding walks. The structure of the proof is very similar, the only difference being the enforcing gadgets used for the construction of the Planar 3-SAT Problem.

2.3 Polynomial Algorithm for Stack, Queue and Staircase

Since finding the contact map overlap between two self-avoiding walks is NP-hard, Goldman et al. (1999) propose ways to decompose these graphs into simpler special cases and derived polynomial time algorithms for these special cases. Before understanding these special cases, it is necessary to understand three different types of relations between pairs of edges in a contact map.

- *Disjoint Edges:* Let $e_1 = (v_i, v_j)$ and $e_2 = (v_{i'}, v_{j'})$ be two edges in G. Then e_1 and e_2 are *disjoint* if $(i, j) \cap (i', j') = \emptyset$.

- *Containing Edges:* e_1 *contains* e_2 if $(i, j) \supset (i', j')$.

- *Overlapping Edges:* Two edges e_1 and e_2 *overlap* each other if $(i, j) \cap (i', j') \neq \emptyset$ and one edge does not contain the other (unless an end point is shared).

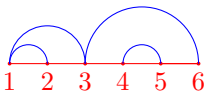

Figure 2.1: Stack

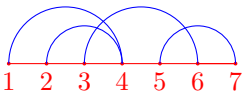

Figure 2.2: Queue

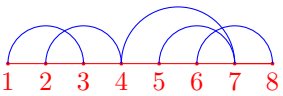

Figure 2.3: Staircase

Based on the above relations, Goldman et al. (1999) propose three simpler graphs as follows:

- *Stack:* A stack is defined as the contact map (V, E) where if e_1 and e_2 are two edges in the contact map then either they contain one another, are disjoint or overlap at one end point.

- *Queue:* A queue is defined as the contact map (V, E) where if e_1 and e_2 are two edges in the contact map then they do not contain one another unless they share one end-point.

- *Staircase:* A staircase is a special case of queue in which there are sets of mutually overlapping intervals such that either no two intervals in the different sets meet, or at most two of them overlap at an end-point.

These graphs are illustrated in Figs. 2.1, 2.2 and 2.3. The proofs corresponding to the computation of contact map overlap for all of these special cases have similar structure and use dynamic programming. Some noteworthy results are presented in the following sections.

2.3.1 $O(n^4)$ algorithm for maximum CMO of 2-stack and degree-2 contact map

Let $\text{co}(G_1, G_2)$ denote the overlap between G_1, a degree-2 stack of n_1 vertices and G_2, a degree-2 contact map of n_2 vertices. Now, since it is tabulated using dynamic programming, the subproblems are formed by computing the contact overlap of subgraphs like $G_{1(v_a,v_b)}$ and $G_{2(u_c,u_d)}$ where $v_1 \leq v_a < v_b \leq v_{n_1}$ and $1 \leq u_c < u_d \leq u_{n_2}$. There are few rules that are followed for computing the contact overlap of these two subgraphs recursively. The first one is that if two edges meet v_b(or u_d) then one must omit the lower first coordinate in each case. These are denoted by $G_{1_{(\hat{v_a},v_b)}}$ or $G_{2_{(\hat{u_c},u_d)}}$. Another rule is that an edge with the lowest end-point in one graph is mapped to the edge with the lowest endpoint in the other. These graphs can be denoted as $\text{co}(G_{1(v_a,v_b)}, G_{2(u_c,u_d)})^l$. Similarly, edges with the highest end point will be mapped to one another. These graphs are denoted as $\text{co}(G_{1(v_a,v_b)}, G_{2(u_c,u_d)})^h$. If both the edges are mapped, then the graph is denoted by $\text{co}(G_{1(v_a,v_b)}, G_{2(u_c,u_d)})^{lh}$. The optimum contact map can be computed as follows:

$$\text{co}(G_1, G_2) = \max\{\text{co}(G_{1(v_1,v_{n_1})}, G_{2(u_1,u_{n_2})})^h, \text{co}(G_{1_{(v_1,\hat{v}_{n_1})}}, G_{2(u_1,u_{n_2})})^h, \text{co}(G_{1(v_1,v_{n_1})}, \cap$$
$$G_{2_{(u_1,\hat{u}_{n_2})}})^h, \text{co}(G_{1_{(v_1,\hat{v}_{n_1})}}, G_{2_{(u_1,\hat{u}_{n_2})}})^h, \text{co}(G_{1(v_1,v_{(n_1-1)})}, G_{2(u_1,u_{n_2})}), \text{co}(G_{1(v_1,v_{n_1})}, G_{2(u_1,u_{(n_2-1)})}\} \quad (2.1)$$

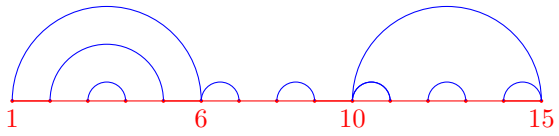

Figure 2.4: $G_{1(1,15)}$

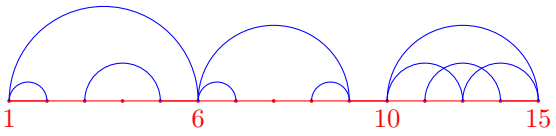

Figure 2.5: $G_{2(1,15)}$

Let us consider two subgraphs in Figs. 2.4 and 2.5. Recursively, the contact map of the two are given as:

$$\text{co}(G_{1(1,15)}, G_{2(1,15)})^{lh} = 2 + \text{co}(G_{1(2,5)}, G_{2(2,5)}) + \max\{\text{co}(G_{1(6,10)}, G_{2(6,11)})^l, \text{co}(G_{1(7,10)}, G_{2(7,11)})\} \cap$$
$$+ \max\{\text{co}(G_{1(10,14)}, G_{2(11,14)})^l, \text{co}(G_{1(11,15)}, G_{2(12,15)})^h, \text{co}(G_{1(11,14)}, G_{2(12,14)})\} \quad (2.2)$$

Here, one can see that node 1 and node 15 of each graph are matched for which 2 is already counted in the calculation. The remaining calculation is for the intermediate edges. One subproblem in such setting is calculation of the number of matchings between nodes 2 and 5 of each graph. The next part of the equation computes the number of edges matching between node 6 and 10 in graph $G_{1(1,15)}$ and 6 and 11 in graph $G_{2(1,15)}$. It would take the maximum number of edges matched in two scenarios - one would be if the edge with lowest end-point at node 6 matches with the lowest end point at node 6 and the other would be to calculate the total number of edges matched from node 7 to node 10 in one and from node 7 to node 11 in the other. Matching at node 10 and 11 is excluded because there are no incoming edges at them. The third part of the calculation also follows the same order.

If G_1 is of size n, then from Eq. 2.1, it can be seen that the total number of entries in the table is $O(n^4)$ and each table entry takes $O(1)$ time to compute. So the total time taken to compute all the entries is $O(n^4)$.

2.3.2 $O(n^3)$ algorithm for maximum CMO of 2-staircase and degree-2 contact map

Considering G_1 to be a degree-2 staircase containing n_1 vertices and G_2 to be any degree-2 graph containing n_2 vertices, the computation of $\text{co}(G_1, G_2)$ is done again using a dynamic programming method. Firstly, the

edges in G_1 and G_2 are numbered according to the ascending order of their right end points. If there are two edges which share the same right end point, the edge with the higher lowest end point is given higher number. A table is constructed with these two graphs having these 3 indices - an edge e_i from G_1, an edge f_j from G_2 and the next entry is either a blank denoted by "-" or contains a higher edge $f_{j'}$ of G_2 whose interval overlaps edge f_j at more than one points. Thus the table entries contain a contact overlap between a subgraph of G_1 denoted by $G_{1_{e_i}}$ which consists of all edges smaller than or equal to e_i and a subgraph of G_2 denoted by $G_{2_{(f_j,f_{j'})}}$ which consists of all edges smaller than or equal to f_j. There is a constraint of the above contact map which states that all edges of $G_{2_{(f_j,f_{j'})}}$ that overlap f_j also overlaps $f_{j'}$ at more than one point. It is reasonable to say now that $f_{j'}$ being the edge which overlaps f_j at more than one point also matches an edge overlapping e_i. Also like the proof in the earlier case, if the highest endpoints of e_i and f_j map to each other, the corresponding contact map overlap is denoted by $\text{co}(G_{1_{e_i}}, G_{2_{(f_j,f_{j'})}})^h$.

Assuming that G_1 and G_2 contain n_1 and n_2 number of edges, the contact overlap between the two graphs is given by:

$$\text{co}(G_1, G_2) = \max\{\text{co}(G_{1_{n_1}}, G_{2_{(n_2,-)}})^h, \text{co}(G_{1_{(n_1-1)}}, G_{2_{(n_2,-)}}), \text{co}(G_{1_{n_1}}, G_{2_{(n_2-1,-)}})\} \tag{2.3}$$

The subproblem $\text{co}(G_{1_{n_{e_1}}}, G_{2_{(n_{e_2},-)}})^h$ here which would recursively operate is also calculated similarly to the ones in the previous algorithm. The edges would be selected as those edges which overlap e_i or f_j(in the staircase it would always be $e_(i-1)$) and the third entry would either be absent or would be the edge that overlaps e_i or f'_j at more than one point. Now, considering all the columns in the table entries, the total number of entries are $O(n^3)$(if n is the number of vertices in G_1) and each table entry takes $O(1)$ time to compute. So the total time taken to compute all the entries is $O(n^3)$.

Now, any RNA structure (except one) is known to be decomposable into two degree-1 stacks (Goldman et al., 1999). Any of these degree-1 stacks will have half of the edges that must be in the optimal alignment. So when this contact map of the RNA structure is optimized with any other contact map using the algorithm used for stack it can be guaranteed that the output will have at least half of edges mapped in the optimal solution. Thus one obtains a 2-approximation algorithm for constructing a contact map between two RNA structures. In the case of a queue, finding the contact map overlap between a queue and another contact map is NP-complete. The approximation algorithm for queues are not shown separately because each queue can be decomposed into two staircases and following the same logic described above, there is a 2-approximation algorithm for computing the contact map overlap of a queue and another contact map.

The best results for approximation algorithm are obtained for a special case of contact map called augmented staircase. An *augmented staircase* is a contact map which can be decomposed into a staircase and a stack with the constraint that for every stack edge and for every staircase edge, the interval formed by the edges are disjoint, overlap at an end point or the interval formed by the staircase edge contains the interval formed by the stack edge. The best polynomial results were obtained by computing the contact overlap between an augmented staircase and any degree-2 contact map.

2.3.3 $O(n^6)$ algorithm for maximum CMO of augmented 2-staircase and degree-2 contact map

This algorithm builds the same way as the staircase algorithm and uses the stack algorithm as a subroutine. Let G_1 be a degree-2 augmented staircase of n_1 vertices and G_2 any degree-2 graph of n_2 vertices. Here edges in G_1 and G_2 are numbered according to ascending order of their right end points. G_1 can be decomposed into a staircase T and a stack S. The stack algorithm is used to compute the table entries for the comparison between subgraphs of S and subgraphs of G_2. In T, if two edges share the same right end-point, the edge with the higher left end point is numbered higher. The table here has four entries: $e = (v_i, v_j)$ from T, $f = (u_k, u_l)$ from G_2. The other two table entries are either empty or a higher edge e' of T which overlaps with e at more than one point and an edge f' of G_2 whose interval overlaps with f at more than one point. The above table entry contains the constrained contact overlap between the subgraph G_1 denoted $G_{1_{(e,e')}}$, consisting of all the edges less than or equal to e and excluding the edges in S which are in the interval formed by e and e', and subgraph of G_2 denoted by $G_{2_{(f,f')}}$, consisting of all edges less than or equal to f.

Here the constraint is that e must map to f and the mapped edges of $G_{2(f,f')}$ which overlap f overlaps f' at more than a point. The contact overlap of G_1 and G_2 is thus given by:

$$\text{co}(G_1, G_2) = \max\{\text{co}(S, G), \max_{e \in T, f \in G_2} \{\text{co}(G_{1(e,-)} G_{2(f,-)}) + s(j, l)\}. \tag{2.4}$$

$$where, s(j,l) = \begin{cases} \max\{co(S_{(j,n_1)}, G_{2(l,n_2)})^l, co(S_{(j+1,n_1)}, G_{2(l+1,n_2)})\}, \\ \text{edges of } S \text{ and } G_2 \text{ meet } j,\ l \text{ as left end points.} \\ co(S_{(j+1,n_1),G_{2(l+1,n_2)}}), \text{otherwise.} \end{cases} \tag{2.5}$$

The optimal bijection maps the highest edges of T to an edge of G_2 or there is no such edge. The recursion maps the next highest edge in the subgraph of T to G_2. So, if $G_1$is of size n, the total number of table entries is $O(n^4)$ and each entry takes $O(n^2)$ time to compute. Hence, the total time to compute the contact overlap between an augmented staircase and any degree-2 graph is $O(n^6)$.

2.4 Decomposition Theorem and Approximation Results

In the earlier section, polynomial algorithms were proposed to calculate the maximum contact overlap between the special cases of graphs. In this section, it will be explored how any self-avoiding walk can be decomposed into these simpler graphs thereby facilitating the construction of contact maps for any self-avoiding walk. The first result, explained below, states that any self avoiding walk can be decomposed into two stacks and one queue.

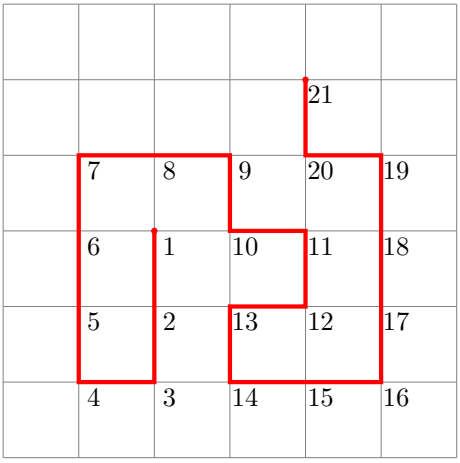

Figure 2.6: Self-avoiding Walk

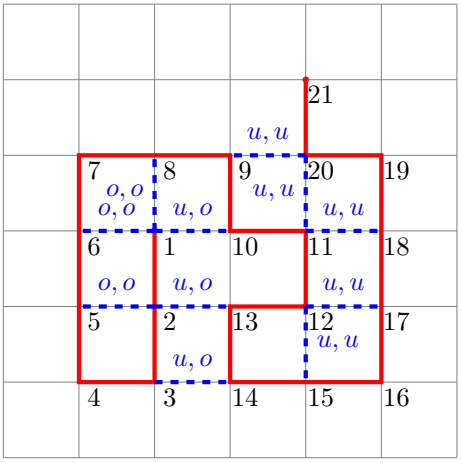

Figure 2.7: Edges marked with Over or Under

Let us consider the self avoiding walk in Fig. 2.6 as an example. Intuitively, a self-avoiding walk is said to have two "sides". Contacts or edges that corresponds to the two sides (Over or Under) are said to form the two stacks and the contacts forming on differing sides (Over and Under) are said to form the queue. Each edge in the above self-avoiding walk is labeled O or U according to the following scheme:

- For non-walk edges adjacent to vertex 1:
  - Among the two edges perpendicular to edge $\{1, 2\}$, one is labeled O and the other U.
  - The remaining edge, if in the contact map, should be assigned the same label with which it forms a lattice square with the edges of the walk. Else it is labeled randomly.

- For non-walk edges adjacent to vertex i where $2 \leq i \leq (n-1)$ the following rule is adopted:
  - If the walk edge from vertex $(i-1)$ to vertex i is in the same straight line, then at least one of the non-walk edges adjacent to i and perpendicular to the edge $\{(i-1), i\}$ will be in the same lattice square with the edges marked at vertex $(i-1)$. In that case this new edge will have the same label.If that edge is labeled L then the other non-walk edge adjacent to i would be labeled $\{O, U\} \setminus \{L\}$.
  - If i is a corner, the edges adjacent to it will share the same label. If any of the adjacent edges forms a lattice square with any edge of $(i-1)$, it will be labeled the same. Else (*i.e.* when $(i-1)$ is also a corner) then edges adjacent to i will be labeled the opposite label assigned to $(i-1)$.
- For non-walk edges adjacent to vertex n:
  - At least one of the non-walk edges adjacent to n and perpendicular to the edge $\{(n-1), n\}$ will be in the same lattice square with the edges marked at vertex $(n-1)$. In that case this new edge will have the same label.If that edge is labeled L then the other non-walk edge adjacent to n would be labeled $\{O, U\} \setminus \{L\}$.
  - If the remaining edge adjacent to vertex n is contained in the contact map, the same label is assigned as whichever of the two previously labeled edges is contained in the closed curve formed by the walk edges and the edge we are considering, else it is labeled arbitrarily.

Using the labeling scheme mentioned above, the self-avoiding walk in Fig. 2.6 can be labeled. The edges with two labels are only shown in Fig. 2.7 as they are the edges of the contact map. From Fig. 2.7, it can be seen that there are three types of labeling of the edges with two labels in the graph - $\{O, O\}$, $\{U, U\}$ and $\{O, U\}$. Now we draw three graphs arranging the vertices in a linear scheme. The edges marked with $\{O, O\}$ will form one graph, the edges labeled $\{U, U\}$ the second graph and the edges labeled $\{O, U\}$ the third graph. Fig. 2.8 shows the three graphs. It is clearly seen that two of them are stacks and the third one is a queue, thus proving the decomposition theorem.

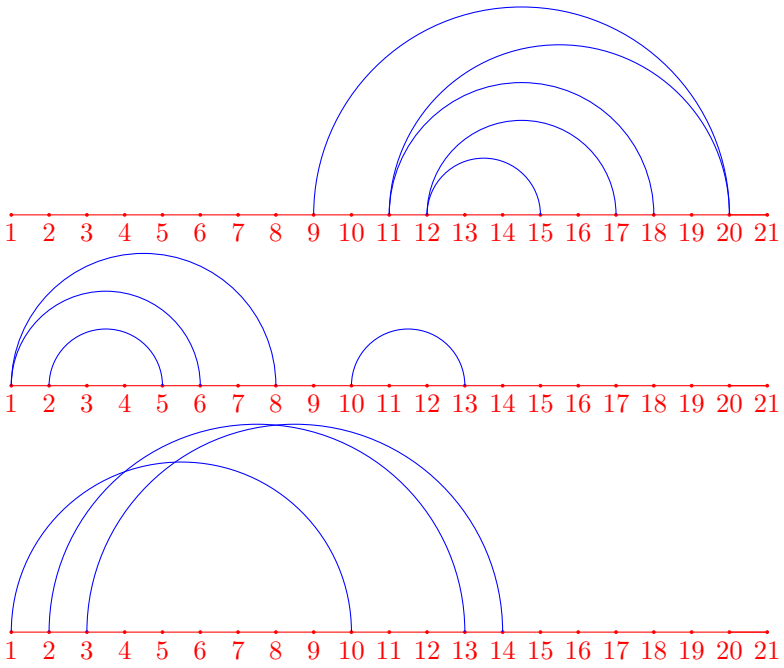

Figure 2.8: Two Stacks and the queue

The decomposition theorem is considered an important step in formulating approximation algorithms for contact map overlap of self-avoiding walks. The first result in computing the contact map overlap is a

4-approximation algorithm. Each walk in such setting is decomposed into two stacks and two staircases of maximum degree 2. This is because each walk can be decomposed into two stacks and a queue and each queue can be decomposed into two staircases of maximum degree 2. Polynomial time algorithms for these special cases are used to compute the 4-approximation algorithm. Considering the above decomposition theorem and with some minor modifications, it can further be shown that any contact map of a self-avoiding walk in two dimensional square lattice can be broken into one stack and two augmented staircases. This decomposition theorem is used to formulate the polynomial-time 3-approximation algorithm.

Goldman et al. (1999) also state two very important open problems. One is to find better approximation ratio for contact map overlaps in two-dimensional lattice. The other is to find properties of three-dimensional self-avoiding walks and investigate if any approximation results can be obtained for the same. These two open problems are addressed in Agarwal et al. (2007).

2.5 Improvement of Result in 2-dimensional model

Agarwal et al. (2007) address two open problems from Goldman et al. (1999). Although they do not provide a better approximation algorithm, the running time of the 6-approximation algorithm is $O(n^3 \log n)$ which is a significant improvement from the previous algorithm in two dimensions. The algorithms described in the next two subsections use the same structure of proof from Goldman et al. (1999).

2.5.1 Staircase and CMO

Towards simplifying the problem of calculating the contact map overlap measure, an algorithm for computing the same between a staircase and the graph of a self-voiding walk is presented in this section. One can observe that if $T = (V, E)$ is a 2-staircase graph then E can be decomposed into two subsets E_1 and E_2 in $O(|E|)$ time where $T_1 = (V, E_1)$ and $T_2 = (V, E_2)$ are 1-staircase graphs. To understand why this is possible at all, one can consider assigning the edges corresponding to each of the degree 2 nodes of the original staircase T to sets E_1 and E_2 alternatively (Fig. 2.9 illustrates this formulation). Since a subgraph of a staircase is also a staircase, both $T_1$and T_2, formed by the above process, become degree 1 staircase graphs. Therefore, to measure the contact map similarity between a 2-staircase graph and a general graph, it is sufficient to propose algorithm for computing contact map measure between a 1-staircase graph and a general graph.

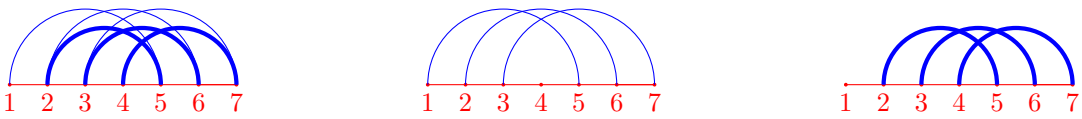

Figure 2.9: A 2-staircase and its decomposition into two 1-staircase

For further description of the algorithm, the concept of "cluster" is important which is defined as a set of mutually overlapping edges. From the definition of a staircase graph, it should be obvious that each 1-degree staircase graph consists of only disjoint clusters of edges. Consider one such 1-degree staircase graph G_1 consisting of clusters $C_1, C_2, \cdots, C_K$ which is compared to a contact map $G_2 = (V_2, E_2)$. Let G_{1i} be the subgraph of G_1 consisting of clusters $C_1, C_2, \cdots, C_i$ and $G_{2(1j)}$ be the subgraph of G_2 consisting of vertices with indices running from 1 to j along with the associated edges. A quantity $D(i, j)$ is defined now to measure the contact map overlap between G_{1i} and $G_{2(1j)}$. The idea is to propose algorithm and measure the complexity for the subgraphs and apply recursion thereafter to compute the similarity measure over the entire graphs.

A consequence of such construction is the following recursive update equation:

$$D(i, j) = \max_{0 \le r \le m_i} \{D(i-1, \Phi(j, r) - 1)\} + r, \tag{2.6}$$

where $\Phi(j, r)$ denotes the largest integer h such that $G_{2(hj)}$ contains a cluster of size r as its subgraph and m_i is the number of edges in cluster C_i. Let h^* denote the largest integer such that $G_{2(h^*j)}$ contains r^* number of edges obtained from a successful maximal mapping f^* between C_i and $G_{2(1j)}$. Then, intuitively Eq. 2.6 implies that once one has computed $D(i-1, h^*-1)$, the maximum overlap measure between $G_{1(i-1)}$ and $G_{2((h^*-1)j)}$, one can compute $D(i, j)$ by simply solving a maximization problem, assuming $\Phi(j, r)$ is pre-computed. One can see that each step of such recursion takes $O(n_{1i})$ time and hence the total computation time is $O(\sum_{n_{1i}} n_2)$ or $O(n_1 n_2)$.

Pre-computation of $\Phi(j, r)$ can be performed using the following recursive update:

$$\Phi(j, r) = \max\{\Phi(j-1, r), \max_{e=(i,j)\in E} \tilde{\Phi}(e, r)\}. \tag{2.7}$$

Here $\tilde{\Phi}(e, r)$ is an auxiliary function corresponding to edge $e \in G_2$ and $r \leq n_2$ such that $\tilde{\Phi}(e, r)$ is the largest integer h so that $G_{2(}hj)$ contains a cluster of size r as its subgraph with e being the rightmost edge of the cluster. Computation of each of such auxiliary functions can be performed in $O(n_2 \log n_2)$ time according to arguments given in Knuth (1973). Since G_2 has $O(n_2)$ edges, total time spent in such computation is $O({n_2}^2 \log n_2)$. Combined everything together, the contact map measure between G_1 and G_2 can be computed in $O(n_1 n_2 + {n_2}^2 \log n_2)$ time.

2.5.2 Stack and CMO

Similar to the construction of the algorithm in the last section, Agarwal et al. (2007) show that the contact map overlap between a 2-stack and any two dimensional graph can be constructed in $O(n_1 {n_2}^2 \log n_1)$ time where n_1 is the number of vertices in the 2-stack and n_2 is the number of vertices in the two dimensional graph. To illustrate the same, the stack has been represented as a tree where each node corresponds to an edge of the tree and a node e' can be considered as the child node of e when e contains e' (Agarwal et al., 2007).

Figure 2.10: Stack and its tree representation

The authors show the results for a 1-stack and any two dimensional graph and then the results are extended to compute the overlap between a 2-stack and graph. Let $G_1 = (V_1, E_1)$ be the 1-stack of size n_1 and $G_2 = (V_2, E_2)$ a contact map of size n_2. Further, let $e \in E_1$ be the rightmost edge in G_1 with the highest right end-point and $e' \in E_2$ be a similar edge in G_2. Since G_1 is a 1-stack, e is not contained in any other edge, so e and e' must match each other. If (v_l, v_j) be the end points of such an edge e in G_1 and $(v_{l'}, v_{j'})$ be the edge of e' in G_2, then the problem can be divided into two subproblems where (v_i, v_l) and (v_l, v_j) are the starting and ending vertices of the two subgraphs of G_1, and $(v_{i'}, v_{l'})$ and $(v_{l'}, v_{j'})$ are the starting and ending vertices of the two subgraphs of G_2. Hence, the maximum contact overlap can be computed by the same dynamic programming technique used by Goldman et al. (1999) where at each step the graph is divided into subgraphs with respect to the rightmost edge. Thus the quantity $D(i, j, i', j')$, the

optimal match between $v_i, \cdots, v_j$ and $v'_i, \cdots, v'_j$, can be given by:

$$D(i,j,i',j') = \max\{D(i,j-1,i',j'), D(i,j,i',j'-1), (D(i,l-1,i',l'-1) + D(l,j-1,l',j'-1)+1)\}, \quad (2.8)$$

where $(v_l, v_j) \in E_1$ and $(v'_l, v'_j) \in E_2$. The time taken to compute this is $O({n_1}^2 {n_2}^2)$. In the equation, the last 1 is when the rightmost edge of each have been mapped. This is known as the "right matching" algorithm. Similarly the algorithm which matches the left edges is known as the "left matching" algorithm, the running time of which is still $O({n_2}^4)$. Using a tree decomposition theorem proposed by Klein (1998), Agarwal et al. (2007) show that if the process is interleaved, *i.e.* at some points the rightmost edges are matched and at others the leftmost edges are matched thereby meeting in the middle, then the total number of edges that need to be visited is $O(n_1 \log n_1)$. This is where the tree structure of the stack becomes useful. These edges need to be mapped to all the edges of V_2 requiring $O({n_2}^2)$ time. Hence, the total time to compute D is $O(n_1 {n_2}^2 \log n_1)$. The vertices which have degree 2 are also handled in the same time bound.

Goldman et al. (1999) show that any two-dimensional contact map can be decomposed into two 2-stacks and two 2-staircase. Further, each 2-staircase can be decomposed into two 1-staircase in linear time. Considering two-dimensional contact maps G_1 and G_2 of size n each, G_1 can be decomposed into six subgraphs and then maximum contact overlap can be calculated between the set of these 6 subgraphs and G_2 rendering a 6-approximation algorithm. The maximum time taken is $O(n^3 \log n)$.

2.6 CMO in 3-dimension

(Agarwal et al., 2007) present the first result of computing the contact map overlap between two graphs in the case of 3-dimensional self-avoiding walks. Let $G = (V, E)$ be any ordered graph with an ordered set of vertices $V = (v_1, v_2, \cdots, v_n)$ and E be the edges arranged in increasing order of their left end-points *i.e.* $E = \{e_1 = (v_{s_1}, v_{t_1}), \cdots, e_u = (v_{s_u}, v_{t_u})\}$, where $s_1 \leq \cdots \leq s_u$ are the starting vertices of the edges. Further, let $\tau = (t_1, \cdots, t_u)$ and σ be defined as the smallest integer k such that τ is decomposed into k monotonically increasing or decreasing subsequences. The decomposition of τ into k monotonically increasing or decreasing subsequences takes place in $O(n \log n)$ time according to the algorithm suggested in Knuth (1973). Since G is a constant graph of degree n, length of τ is $O(n)$ and the number of iterations to create this subsequence is at most $O(\sqrt{n})$ (Erds P, 1935).

Let the subsequence extracted at the k^{th} iteration be given by $\tau_k = (t_{i1}, ..., t_{il})$ and the subgraph formed by these edges in τ_k be called T_k where the edge set is given by $E_k = \{(s_{i1}, t_{i1}), \cdots, (s_{il}, t_{il})\}$. It can be shown that either T_k is a stack (in a decreasing subsequence) or a queue (in an increasing subsequence). Two edges $e_i = (v_{s_i}, v_{t_i})$ and $e_j = (v_{s_j}, v_{t_j})$ are so taken that $t_i \in \tau_k$ at index s_i and $t_j \in \tau_k$ at index s_j. Without loss of generality, for any subsequence one can consider $t_i < t_j$. If τ_k is an increasing subsequence, $s_i < s_j$ and consequently, e_i does not contain e_j and *vice versa* for which τ_k becomes a queue. For a decreasing subsequence $s_i > s_j$ and s_j, the left index point of e_j, is less than s_i, the left index point of e_i, and t_j, the right index point of e_j, is more than t_i, the right index point of e_i. Thus e_j contains e_i and hence T_k is a stack.

Now at any iteration, τ_k is either a stack or a queue. Since each queue can be divided into two staircases, the total number of subgraphs is at most $O(\sqrt{n})$. Again, since each subgraph is computed in $O(n \log n)$ time and there are σ of such subgraphs, the total running time is $O(\sigma n \log n)$.

Analogous to the contact map overlap in 2-dimensional walks, to construct the overlap between two contact maps G_1 and G_2 of two 3-dimensional walks, either G_1 or G_2 is decomposed into σ stacks and staircases and then contact map overlap of each can be computed in polynomial time. Contact map overlap will be computed between these σ subgraphs and G_2, achieving a σ-approximation algorithm. Summing up the running time of matching each of the smaller graphs to the other, the total running time is $O(n^3 \log n)$.

Chapter 3

Exact Algorithms for Maximum CMO

Though NP-hard problems are not solvable in polynomial time, they can be solved through exhaustive search. It is when the size of the search space grows, the running time becomes astronomically large or memory becomes a bottleneck. However, for some special cases, it is possible to design algorithms that are faster than the exhaustive search. These algorithms are called "Exact" algorithms. There exists four such exact algorithms for the maximum CMO problem where the branch-and-bound framework is adopted to ensure global optimality. The four algorithms are – B & Cut (Carr et al., 2000), Clique (Strickland et al., 2005), LAGR (Caprara et al., 2004) and CMOS (Xie and Sahinidis, 2007). The algorithm proposed by Andonov et al. (2011) and discussed extensively in this report outperforms the other exact algorithms and successfully reproduces the results of previous algorithms in much less time. The sections below discuss the formulation of the mathematical model and also the structure and proofs of the algorithm in Andonov et al. (2011).

3.1 Integer Programming Model for CMO

The framework proposed by Andonov et al. (2011) represents the CMO problem as an integer programming problem. When tested on a widely used benchmark, the Skolnick set (Godzik and Skolnick, 1994), the proposed algorithms for solving the optimization problem outperforms other existing exact algorithms both in terms of time efficiency and quality of bounds. To describe the formulation further, let us consider the contact maps of two proteins P_1 and P_2 given by $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where n_1 is the number of vertices in G_1 and n_2 is the number of vertices in G_2. For convenience, contact maps of both the proteins will be represented by the index m that takes value 1 for P_1 and value 2 for P_2. The right and left neighbour of any vertex i in the m^{th} protein is given by $\delta_m^+ = \{j | j > i, (i,j) \in E_m)\}$ and $\delta_m^- = \{j | j < i, (j,i) \in E_m)\}$ respectively. For every edge $(i,j) \in E_1$ mapped to $(k,l) \in E_2$, a non-crossing alignment happens when $i < j$ and $k < l$. Feasible alignments of two proteins P_1 and P_2 are given by non-crossing matchings in the complete bipartite graph B with a vertex set $V_1 \cup V_2$. For every such non-crossing alignment, a weight w_{ikjl} is introduced according to the following rule:

$$w_{ikjl} = \begin{cases} 1, & \text{if } (i,j) \in E_1 \text{ and } (k,l) \in E_2, \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

If M represents the complete set of non-crossing matching in B, then $w(M)$ is the sum of all the weights over edges in M and the CMO problem is nothing other than maximizing $w(M)$.

Now, let us consider an interesting transformation of the problem also studied extensively in Andonov et al. (2004, 2008). The edges of B are mapped onto a gird of size $n_1 \times n_2$ which is denoted by $B' = (V', E')$. The mapping abides by the following rule – a vertex $(i,k) \in V'$ if there exists an edge (i,k) in B and an arc $(i,k)(j,l) \in E'$ if $w_{ikjl} = 1$. A bipartite graph B and its corresponding grid graph transformation are illustrated in Fig. 3.1 (Andonov et al., 2011). In Fig. 3.1, only the edges of the considered matching

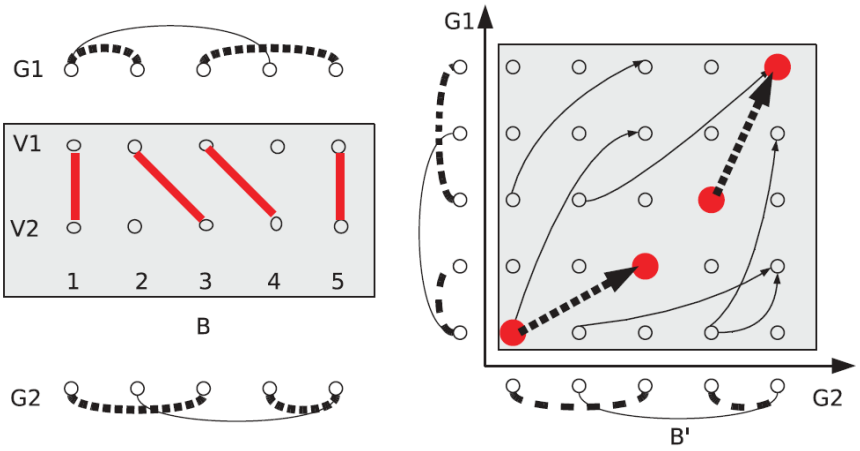

Figure 3.1: Bipartite and Grid Graph Representation

are shown. Further, let's consider "feasible path" in this grid which is defined as an arbitrary sequence of vertices whose coordinates are arranged in increasing order. For example, in Fig. 3.1, a feasible path is a path connecting the vertices $(1,1)$, $(2,3)$, $(3,4)$ and $(5,5)$. Since, $M = \{(1,1)(2,3),(3,4)(5,5)\}$, according to the update rule given in Eq. 3.1, $w(M) = 2$. One can see that the arcs $(1,1)(2,3)$ and $(3,4)(5,5)$ are activated only in Fig. 3.1 which contribute towards making $w(M) = 2$. Therefore, in the transformed representation B', one can see that the objective is to maximize the number of such arcs whose vertex set forms the feasible path.

By assigning a binary variable x_{ik} to each vertex $(i,k) \in B'$ and a binary variable y_{ikjl} to each arc $(i,k)(j,l) \in B'$ and considering X to be a feasible path in B', the maximum CMO problem can be represented as an integer programming problem as follows:

$$v(\mathbf{x},\mathbf{y}) = \max_{\mathbf{x},\mathbf{y}} \sum_{(ik)(jl)\in E'} y_{ikjl}, \text{ subject to,} \tag{3.2}$$

$$x_{ik} \geq \sum_{l\in\delta_2^+(k)} y_{ikjl}, j \in \delta_1^+(i), i \in [1, n_1 - 1], k \in [1, n_2 - 1], \tag{3.3}$$

$$x_{ik} \geq \sum_{l\in\delta_2^-(k)} y_{ikjl}, j \in \delta_1^-(i), i \in [2, n_1], k \in [2, n_2], \tag{3.4}$$

$$x_{ik} \geq \sum_{l\in\delta_1^+(i)} y_{ikjl}, l \in \delta_2^+(k), i \in [1, n_1 - 1], k \in [1, n_2 - 1], \tag{3.5}$$

$$x_{ik} \geq \sum_{j\in\delta_2^-(i)} y_{ikjl}, l \in \delta_2^-(k), i \in [2, n_1], k \in [2, n_2], \tag{3.6}$$

$$\{x_{ik}\} \in X, \tag{3.7}$$

$$\sum_{l=1}^{k} x_{il} + \sum_{j=1}^{i-1} x_{jk} \leq 1, i \in [1, n_1], k \in [1, n_2]. \tag{3.8}$$

Here, $\mathbf{x} = \{x_{ik}\}$ and $\mathbf{y} = \{y_{ikjl}\}$. A feasible path X is constructed by taking a complete grid graph and then pruning the edges based on the constraints described above. To understand the first constraint (Eq. 3.3) in details, let us consider a CMO problem where the i^{th} residue from one protein is mapped to the k^{th} residue of the other protein. Considering that $(i,j) \in E_1$ and the fact that CMO is an order preserving bijection,

j can only be mapped to one point in the the second contact map *i.e.*, l in the second contact map will be unique. Constraint 3.5 is similar to the constraint 3.3 but corresponds to the reverse order. If there is an edge from k to i in the bipartite graph (*i.e* k^{th} residue from the second protein is mapped to i^{th} residue in protein 1)and if $(k,l) \in E_2$ is an edge in the second contact map, then j(the point l can be mapped to) of the first contact map should also be unique. Here the summation signifies that for all values of l, only one can be 1,the remaining has to be 0. Similarly constraints 3.5 and 3.6 are constructed. The sets of the right and left neighbours are useful to formulate the same. Again weight w_{ikjl} is one when the matching is non crossing, so the arc $(i,k)(j,l)$ is activated i.e. $y_{ikjl} = 1$ iff $x_{ik} = 1$ and $x_{jl} = 1$.

Constraint 3.8 can be understood with the following example. Let i be any residue in P_1. Now if i is mapped to k in P_2,then x_{ik} is 1 and it satisfies the constraint. If i is not mapped to k, then either i will be mapped to any element from 1 to $(k-1)$ or k will be mapped to any element from 1 to $(i-1)$. This also reiterates the fact that there is only one bijective mapping between a residue in one protein to the other in another protein. Again, this mapping should be non-crossing, thereby reassuring order-preserving bijection.

3.2 Solution of Integer Programming Model for CMO

The objective function in Eq. 3.2 can be optimized using different techniques. Earlier approaches (Carr et al., 2000; Caprara et al., 2004) solve the problem using only branch-and-bound technique (Papadimitriou and Steiglitz, 1982; Bertsimas and Tsitsiklis, 1997) or branch-and-cut technique the latter of which can be considered as an *integer constrained relaxation* method for integer programming (Mitchell, 1998). However, as suggested in the optimization literature (Bertsekas, 1999; Bertsimas and Tsitsiklis, 1997), significant computation advantage can be gained by performing *Lagrangian relaxation* as well. In such relaxation, first a feasible set is formulated where the *Lagrangian* is optimized *w.r.t* the *primal variables* and the dual objective is optimized thereafter yielding a lower (for minimization) or upper bound (for maximization) to the original objective. The duality gap can further be improved by proposing better constraint sets (Bertsekas, 1999; Bertsimas and Tsitsiklis, 1997). Additionally, in case of a non-zero duality gap, one can employ branch-and-bound algorithm on the smaller feasible set obtained from the Lagrangian relaxation. In this section, first a tighter constrain set, proposed by Andonov et al. (2011), is presented which is then followed by the Lagrangian relaxation and an application of the branch-and-bound algorithm.

3.2.1 Finding a Tighter Constrain Set

Tightening the solution space may improve the bound of the proposed Lagrangian (Achterberg et al., 2005; Bertsekas, 1999). To increase the constrain set for a tighter relaxation, the constraint 3.4 and 3.6 are replaced by the following two constraints:

$$x_{ik} \geq \sum_{(r,s)\in row_{ik}(j)} y_{rsik}, j \in \delta_1^+(i), i \in [1,n_1], k \in [1,n_2] \tag{3.9}$$

$$x_{ik} \geq \sum_{(r,s)\in col_{ik}(l)} y_{ikjl}, l \in \delta_2^-(k), i \in [2,n_1], k \in [2,n_2] \tag{3.10}$$

The construction of $row_{ik}(j)$ and $col_{ik}(l)$ in Eqs. 3.9 and 3.10 can be explained with the Fig. 3.2. The grey area in each of the images is the rectangle formed by $\delta_1^-(i)$ rows and $\delta_2^-(k)$ colums for any vertex (i,k) in the graph. Let $(i_1, i_2, \cdots, i_s)$ and $(k_1, k_2, \cdots, k_t)$ be the ordered set of vertices in $\delta_1^-(i)$ and $\delta_2^-(k)$. Considering the vertices in the l^{th} column to be the pairwise crossing matching, at most one of the points in the vertices of the column will satisfy Eq. 3.6. The solution remains the same even if the following sets are added to the vertices in the l^{th} column:$(i_s, k_1), (i_s, k_2), \cdots, (i_s, k_{l-1})$ and $(i_1, k_{l+1}, i_1, k_{l+2}, \cdots, i_1, k_t)$. The union of these three sets constructs $col_{ik}(l)$. $row_{ik}(j)$ is constructed in a similar fashion.

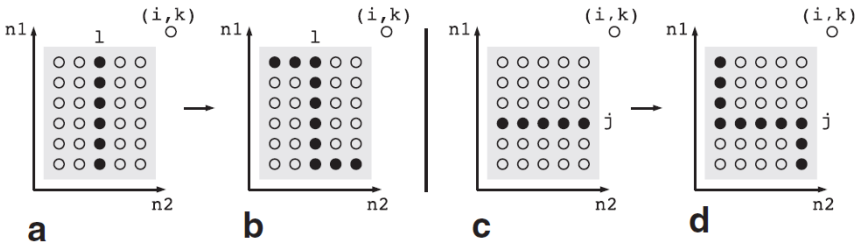

Figure 3.2: Construction of tighter constraint set

3.2.2 Lagrangian Relaxation

The Lagrangian of the problem in Eq. 3.2 is formed as follows:

$$\mathcal{L}(\mathbf{x},\mathbf{y},\boldsymbol{\lambda}) = \sum_{(ik)(jl)\in E'} y_{ikjl} + \sum_{i,k,j\in\delta_1^{-1}(i)} \lambda^h_{ikj}\left(x_{ik} - \sum_{(r,s)\in \mathrm{row}_{ik}(j)} y_{rsik}\right) \tag{3.11}$$

$$+ \sum_{i,k,l\in\delta_2^{-1}(k)} \lambda^v_{ikl}\left(x_{ik} - \sum_{(r,s)\in \mathrm{col}_{ik}(j)} y_{rsik}\right) \tag{3.12}$$

with the constraints 3.3,3.5,3.7,3.8 and $\boldsymbol{\lambda} = \{\{\lambda^h_{ikj}\},\{\lambda^v_{ikl}\}\} \geq \mathbf{0}$. The maximization over the primal variables $\mathbf{x},\mathbf{y}$ can be performed in $O(|V'|+|E'|)$ time and this becomes obvious if the objective in Eq. 3.11 is observed more carefully. One can see that the objective decomposes over the primal variables and essentially there are as many subproblems as there are number of primal variables in the objective, each of which can be solved independently. Moreover, two different sets of sub-problems can be defined. The first one corresponds to finding the set of best *local variable* y_{ijkl} pertaining to each vertex $(i,k) \in V'$ that respects the constrains. The second one corresponds to finding the best values of the set of *global variables* $\{x_{ik}\}$'s. Simple dynamic programming based computation shows a run time complexity of $O(|V'|+|E'|)$.

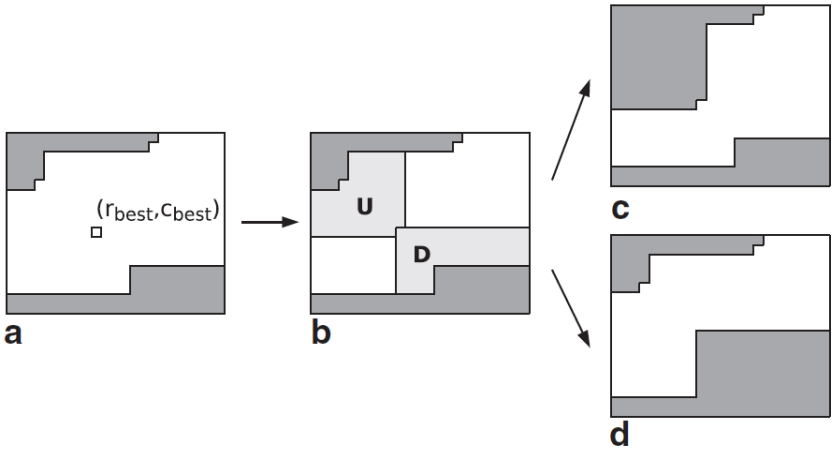

Figure 3.3: Branch-and-bound Splitting Strategy

A dual objective function $\mu(\boldsymbol{\lambda})$ is obtained afterwards by maximizing 3.11 over the primal variables. The dual objective, however, has plenty of variables $\boldsymbol{\lambda}$ and hence sub-gradient descent (Held et al., 1974; Duchi

et al., 2011) with adaptive step-size regulation is used. However, the bound so obtained is not tight and there might exist duality gap either because of problem formulation or from the approximations used in the sub-gradient optimization. To make the solution even better, branch-and-bound algorithm is used as discussed in the next section.

3.2.3 Branch-and-bound Algorithm

A branch-and-bound algorithm works by enumerating over all possible solution sets and maintaining a tree structure to prune the search space based on the value of the objective function. Parts of the tree where the objective function takes bigger value (in case of maximization), are preferred over other parts of the tree. A node in the tree in the current setting is given by by n_2 couples (b_k, t_k) for $k_2 \in [1, n2]$ which define the candidate vertex set $Cand$ (the white area in-between the two broken lines on Fig. 3.3). A vertex (j, l) of the graph B' belongs to $Cand$ if $b_l \leq j \leq t_l$. For any vertex $(j, l) \in Cand$, two sets are created: $U(j, l) = \{(i, k) \in Cand : (i, k) = (j, l), i \geq j, k \leq l\}$ and $D(j, l) = f(i, k) \in Cand : i \leq j, k \geq l\}$. By definition, if a feasible path has a vertex in $U(j, l)$, it cannot have one in $D(j, l)$, and *vice versa*. Let $(rbest, cbest)$ be the $\arg\max_{(j,l)\in Cand} \min(\|D(j, l)\|, \|U(j, l)\|)$. Now, the two descendants of the current node are obtained by discarding from its feasible set the vertices belonging to the two respective domains $U(rbest, cbest)$ and $D(rbest, cbest)$ (refer to Fig. 3.3). The goal of this strategy is twofold: to create descendants that are balanced in sense of feasible set size and to reduce maximally the parent nodes feasible set.

Chapter 4

Critical Analysis

This chapter provides a critical analysis of comparing protein structures using CMO and also discusses possible extensions and research directions.

4.1 Advantages of Using CMO

4.1.1 CMO vs RMSD

Two most widely accepted scoring schemes for alignment of protein structures are : *root mean square division* popularly known as RMSD (Kabsch, 1976) and *contact map overlap*(CMO). RMSD is a widely-used measure for quantifying the similarity between two protein structures. It is defined as the distance between two residues in comparing proteins depending on their relative position in space. So the matching process consists of two subtasks - finding mapping of residue from one protein to another and also finding the best rotation and translation such that the RMSD is minimized under this mapping. From the definition, it can be said that although it is fairly easy to compute, it however treats protein structures as a rigid object. But in reality, proteins exhibit dynamic structural fluctuations; hence, keeping the structural aspect of the protein in mind, a need for a measure which takes the larger flexibility into account is needed.

Since contact maps concentrate on the structure of the protein and contains enough information to reconstruct the overall geometry of a protein's structure,they overcome the shortcoming of RMSD as a scoring mechanism. Again in a contact map, two residues are said to be in contact if the distance between them is less than a given threshold. So it is insensitive to changes in distance between residues that are far apart from one another, allowing more flexibility than RMSD.

Overall, since RMSD is a global alignment process, any algorithm which uses RMSD as its scoring scheme, is a very poor indicator of the similarity between two proteins.CMO, on the other hand is robust, takes partial matching into account, is translation invariant and captures the intuitive notion of similarity very well (Agarwal et al., 2007).

4.1.2 CMO vs LCP Problem

When it comes to protein structure alignment literature, two main mathematical frameworks known are the Maximum CMO problem and the LCP (Largest common point set) problem. In the LCP problem, one finds the maximum mapping of the largest cardinality where the RMSD of the matched residues is no more than a given threshold. The advantages of the LCP problem are that it can solve the problem optimally and it is fairly easy to compute. However, despite solving the problem optimally, the running time complexity of the problem is so alarmingly high that it is practically impossible to compute the similarity between large proteins. Again since it uses RMSD as its scoring scheme, it does not include the flexibility of the protein structure in its calculations (Li et al., 2013). This is where CMO poses as an advantage. Since CMO is a

graph-theoretic in nature, it captures the structural aspect of the protein thereby being a better process for aligning two proteins.

4.2 Limitations and Areas of Improvement

4.2.1 CMO Problem vs Spectral Methods

Quiet recently, researchers have started using spectral matching techniques of graphs to find the similarity between two protein 3D structures. The similarity measure between two proteins is a novel characterization technique called projections. Bhattacharya et al. (2006) point out in his paper that an advantage spectral methods have over CMO is that comparison with spectral methods are based on two residues (one from each protein) unlike the CMO process which needs 4 residues(two from each protein). Also the spectral methods scale well with the change of distance parameter between two proteins (Bhattacharya et al., 2006).

4.2.2 Improvement of Upper Bounds in Exact Algorithms

Out of the five exact CMO approaches published, B & Cut (Carr et al., 2000), Clique (Strickland et al., 2005), LAGR (Caprara et al., 2004) and CMOS (Xie and Sahinidis, 2007) and the algorithm by Andonov et al. (2011) commonly known as *A_purva*, numerical results show that the last algorithm outperforms the remaining ones. The bounds derived by Lagrangian relaxation outperforms the other algorithms. It would be interesting to see if there could be reasonable improvement in the upper bounds via a combination of combinatorial approaches and mathematical programming approaches.

4.2.3 Practical Utility of Exact Algorithms

Most of the experiments performed on protein structure similarity are done on structures which have been biologically proved to be similar. Aligning proteins with different structures still are much harder than those with similar structure. In experiments performed by Andonov et al. (2011) it is seen that while the time for aligning similar structures(domains from the same family) varies from 0.02 seconds and 2.14 seconds, the time taken to align dissimilar structures(domains from different families) is from 3.47 seconds to more than 1800 seconds. So the practicality of such exact algorithms are questionable. Experiments in this domain are mostly performed on small to medium sized proteins(e.g. 60 residues) and they have consumed two hours of CPU time on a standard workstation (Xie and Sahinidis, 2007). It would be interesting to see how far these algorithms can be extended to incorporate

4.2.4 Biological Information for 3D-optimal Alignment

One of the major drawbacks in this field of measuring similarity between 3D protein structures is the absence of easily available biological information (Xie and Sahinidis, 2007). This is an important research direction because many instances, although mathematically important for testing the algorithm are not worth aligning from a biological perspective. This fact is reiterated in certain cases where the algorithm has given near optimal solutions but the secondary structure information of them show that while one protein consisted of all helixes, the other had strands. It would be interesting to see if secondary structure information could be incorporated in the construction of algorithms that compute contact map overlap.

4.2.5 Machine Learning for CMO Problem

Machine learning being one of the most recent interesting research domains, one can also solve the maximum CMO problems using concepts like structure prediction. A relevant literature is available over more recent set of publications (Kshirsagar et al., 2013; Ho et al., 2013; Zhang et al., 2005; Wen et al., 2005; Teso et al., 2010; Vullo et al., 2008). Additionally, the discrete optimization problem in maximum CMO can be replaced by a suitable sub-modular optimization problem (Iyer and Bilmes, 2012, 2013; Iyer et al., 2013) which can possibly lead to better run time complexity and bound.

Bibliography

Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. 33(1):42–54, 2005.

Pankaj K. Agarwal, Nabil H. Mustafa, and Yusu Wang. Fast molecular shape matching using contact maps. *Journal of Computational Biology*, 14(2):131–143, 2007.

Rumen Andonov, Stefan Balev, and Nicola Yanev. Protein threading: From mathematical models to parallel implementations. *INFORMS Journal on Computing*, 16(4):393–405, 2004.

Rumen Andonov, Nicola Yanev, and Noël Malod-Dognin. An efficient lagrangian relaxation for the contact map overlap problem. In *Algorithms in Bioinformatics*, pages 162–173. Springer, 2008.

Rumen Andonov, Nol Malod-Dognin, and Nicola Yanev. Maximum contact map overlap revisited. *Journal of Computational Biology*, 18(1):27–41, 2011.

B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *J. Comput. Biol.*, 5(1):27–40, 1998.

D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997.

Sourangshu Bhattacharya, Chiranjib Bhattacharyya, and Nagasuma R Chandra. Projections for fast protein structure retrieval. *BMC bioinformatics*, 7(Suppl 5):S5, 2006.

Tom L. Blundell and Mark S. Johnson. Catching a common fold. *Protein Science*, 2(6):877–883, 1993.

Alberto Caprara, Robert Carr, Sorin Istrail, Giuseppe Lancia, and Brian Walenz. 1001 optimal pdb structure alignments: integer programming methods for finding the maximum contact map overlap. *Journal of Computational Biology*, 11(1):27–52, 2004.

Robert D. Carr, Giuseppe Lancia, Sorin Istrail, and Celera Genomics. Branch-and-cut algorithms for independent set problems: Integrality gap and an application to protein structure alignment. In *SAND Report SAND2000-2171, Sandia National Laboratories, 2000, available at http://infoserve.sandia.gov/sand doc/2000/002171.pdf*, 2000.

F. E. Cohen and M. J. Sternberg. On the prediction of protein structure: The significance of the root-mean-square deviation. *J. Mol. Biol.*, 138(2):321–333, Apr 1980.

Pierluigi Crescenzi, Deborah Goldman, Christos H. Papadimitriou, Antonio Piccolboni, and Mihalis Yannakakis. On the complexity of protein folding. *Journal of Computational Biology*, 5(3):423–465, 1998.

R. Diamond. A note on the rotational superposition problem. *Acta Crystallographica Section A*, 44(2): 211–216, Mar 1988.

R. Diamond. On the multiple simultaneous superposition of molecular structures by rigid body transformations. *Protein Science*, 1(10):1279–1287, 1992.

Ken A. Dill and Justin L. MacCallum. The Protein-Folding Problem, 50 Years On. *Science*, 338(6110): 1042–1046, November 2012.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

Szekeres G. Erds P. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935.

Adam Godzik and Jeffrey Skolnick. Flexible algorithm for direct multiple alignment of protein structures and sequences. *Computer applications in the biosciences: CABIOS*, 10(6):587–596, 1994.

Deborah Goldman, Sorin Istrail Y, and Christos H. Papadimitriou Z. Algorithmic aspects of protein structure similarity. In *In 40th Annual Symposium on Foundations of Computer Science*, pages 512–521, 1999.

Michael Held, Philip Wolfe, and HarlanP. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6(1):62–88, 1974. ISSN 0025-5610.

H. K. Ho, L. Zhang, K. Ramamohanarao, and S. Martin. A survey of machine learning methods for secondary and supersecondary protein structure prediction. *Methods Mol. Biol.*, 932:87–106, 2013.

L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.*, 233: 123–138, 1993.

L. Holm and C. Sander. Mapping the protein universe. *Science*, 273(5275):595–603, August 1996.

Hae-Jin Hu, Yi Pan, Robert W. Harrison, and Phang C. Tai. Protein secondary structure prediction using support vector machine with advanced encoding schemes. In Belur V. Dasarathy, editor, *Data Mining and Knowledge Discovery: Theory, Tools, and Technology*, volume 5433 of *SPIE Proceedings*, pages 80–87. SPIE, 2004.

Andrea Ilari and Carmelinda Savino. Protein structure determination by x-ray crystallography. 452, May 2008.

Rishabh Iyer and Jeff A. Bilmes. Submodular-bregman and the lovász-bregman divergences with applications. In *NIPS*, pages 2942–2950, 2012.

Rishabh Iyer, Stefanie Jegelka, and Jeff Bilmes. Curvature and optimal algorithms for learning and minimizing submodular functions. *CoRR*, abs/1311.2110, 2013.

Rishabh K. Iyer and Jeff A. Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *NIPS*, pages 2436–2444, 2013.

Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

Philip N. Klein. Computing the edit-distance between unrooted ordered trees. In *Proceedings of the 6th Annual European Symposium on Algorithms*, ESA '98, pages 91–102, London, UK, UK, 1998. Springer-Verlag.

D.E Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, Reading,MA, 1973.

Meghana Kshirsagar, Jaime Carbonell, and Judith Klein-Seetharaman. Multitask learning for hostpathogen protein interactions. *Bioinformatics*, 29(13):i217–i226, 2013.

Shuai Cheng Li, Songjian Lu, Xinghua Lu, Daniele Catanzaro, Ramamoorthi Ravi, Russell Schwartz, Beatriz Pontes, Raúl Giráldez, Jesús S Aguilar-Ruiz, Xuefeng Cui, et al. The difficulty of protein structure alignment under the rmsd. *Algorithms for Molecular Biology*, 8(1), 2013.

D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

John E. Mitchell. Branch-and-cut algorithms for integer programming. 1998. URL `http://homepages.rpi.edu/~mitchj/papers/bc_hao.pdf`.

Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

Dawn M Strickland, Earl Barnes, and Joel S Sokol. Optimal protein structure alignment using maximum cliques. *Operations research*, 53(3):389–402, 2005.

W. R. Taylor and C. A. Orengo. A holistic approach to protein structure alignment. *Protein Eng.*, 2(7): 505–519, May 1989.

Stefano Teso, Cristina Risio, Andrea Passerini, and Roberto Battiti. An on/off lattice approach to protein structure prediction from contact maps. In TjeerdM.H. Dijkstra, Evgeni Tsivtsivadze, Elena Marchiori, and Tom Heskes, editors, *Pattern Recognition in Bioinformatics*, volume 6282 of *Lecture Notes in Computer Science*, pages 368–379. Springer Berlin Heidelberg, 2010.

Gerrit Vriend and Chris Sander. Detection of common three-dimensional substructures in proteins. *Proteins: Structure, Function, and Bioinformatics*, 11(1):52–58, 1991.

Alessandro Vullo, Andrea Passerini, Paolo Frasconi, Fabrizio Costa, and Gianluca Pollastri. On the convergence of protein structure and dynamics. statistical learning studies of pseudo folding pathways. In Elena Marchiori and Jason H. Moore, editors, *EvoBIO*, volume 4973 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 2008.

Christopher A. Waudby, Hlne Launay, Lisa D. Cabrita, and John Christodoulou. Protein folding on the ribosome studied using {NMR} spectroscopy. *Progress in Nuclear Magnetic Resonance Spectroscopy*, 74: 57 – 75, 2013.

Zhi-ning Wen, Ke-long Wang, Meng-long Li, Fu-sheng Nie, and Yi Yang. Analyzing functional similarity of protein sequences with discrete wavelet transform. *Comput. Biol. Chem.*, 29(3):220–228, June 2005.

Wei Xie and Nikolaos V Sahinidis. A reduction-based exact algorithm for the contact map overlap problem. *Journal of Computational Biology*, 14(5):637–654, 2007.

Guang-Zheng Zhang, D. S. Huang, Y. P. Zhu, and Y. X. Li. Improving protein secondary structure prediction by using the residue conformational classes. *Pattern Recogn. Lett.*, 26(15):2346–2352, November 2005.

Y. Zhang. Progress and challenges in protein structure prediction. *Curr. Opin. Struct. Biol.*, 18(3):342–348, Jun 2008.

M. Zuker and R. L. Somorjai. The alignment of protein structures in three dimensions. *Bull. Math. Biol.*, 51(1):55–78, 1989.